NASA Contractor Report 187613

(2)

ICASE INTERIM REPORT 18

SINGLE-BLOCK NAVIER-STOKES INTEGRATOR

P. A. Jacobs

NASA Contract No. NAS1-18605
July 1991

INSTITUTE FOR COMPUTER APPLICATIONS IN SCIENCE AND ENGINEERING
NASA Langley Research Center, Hampton, Virginia 23665

Operated by the Universities Space Research Association

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665-5225

ICASE INTERIM REPORTS

ICASE has introduced a new report series to be called ICASE Interim Reports. The series will complement the more familiar blue ICASE reports that have been distributed for many years. The blue reports are intended as preprints of research that has been submitted for publication in either refereed journals or conference proceedings. In general, the green Interim Report will not be submitted for publication, at least not in its printed form. It will be used for research that has reached a certain level of maturity but needs additional refinement, for technical reviews or position statements, for bibliographies, and for computer software. The Interim Reports will receive the same distribution as the ICASE Reports. They will be available upon request in the future, and they may be referenced in other publications.

Robert G. Voigt
Director

SINGLE-BLOCK NAVIER-STOKES INTEGRATOR

P. A. Jacobs[1]
Institute for Computer Applications in Science and Engineering
NASA Langley Research Center
Hampton, VA 23665

ABSTRACT

This report describes a program for the time-integration of the Navier-Stokes equations on a two-dimensional structured mesh. The flow geometry may be either planar or axisymmetric. The unusual features of this program are that it is written in C and makes extensive use of sophisticated data structures to encapsulate the data. The idea of writing the code this way is to make it easier (than traditional FORTRAN codes) to "parallelize" for the Multiple-Instruction-Multiple-Data style of parallel computer.

The integral form of the governing equations are given for cartesian coordinates and then the particular discretization used in the code is described. A derivation of the axisymmetric equations is given in an appendix. The full version of the code describes a flow domain as a set of abutting blocks, each consisting of a *tensor-product* mesh of quadrilateral cells. However, this report considers only the single-block version of the code. The flow field is recorded as cell-average values at cell centres and explicit time stepping is used to update conserved quantities. MUSCL-type interpolation and a three-stage Riemann solver are used to calculate inviscid fluxes across cell faces while central differences (via the divergence theorem) are used to calculate the viscous fluxes. The Riemann solver is suitable for flows with very strong shocks and does not require the entropy fix as applied to the Roe-type solvers. Because the code is intended to be a test-bed for implementation on parallel computers, the coding details are described in some detail.

A set of test problems is also included. These exercise various parts of the code and should be useful for both validation and performance measurements of the (future) parallel implementations.

[1]Research was supported by the National Aeronautics and Space Administration under NASA Contract No. NAS1-18605 while the author was in residence at the Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, VA 23665.

Contents

1 Introduction 5

2 Governing Equations 6
- 2.1 Spatial Discretization and Data Storage 7
- 2.2 Inviscid Flux Calculation 9
  - 2.2.1 Inviscid Boundary Conditions 9
  - 2.2.2 Interpolation of the Interface State 9
  - 2.2.3 Riemann Solver 11
- 2.3 Viscous Flux Calculation 14
  - 2.3.1 Viscous Boundary Conditions 15
- 2.4 Time Stepping 15
- 2.5 Computational Effort 16

3 Test Cases 17
- 3.1 One-dimensional Shock Tube 17
- 3.2 High-Temperature Shock Tube 17
- 3.3 Double Mach Reflection 18
- 3.4 Flat Plate Boundary Layer 19
- 3.5 Inviscid Flow over a Cone 20
- 3.6 Viscous Flow along a Cylinder 21
- 3.7 High Mach Number Flow around a Sphere. 22

4 Concluding Remarks 24

A Axisymmetric Flow Geometry 28
- A.1 Governing Equations in Cartesian Coordinates 29
- A.2 Application to the Axisymmetric Cell 30
- A.3 Treatment of the interfaces at $\theta = \pm\psi$ 31
- A.4 Summary of the Axisymmetric Equations 32

B Approximate Power Routine 34

C Header File 35

Nomenclature, Units

A : (x,y)-plane cell area
$\mathbf{A}$: data structure name
a : local speed of sound, m/s
C_p : coefficient of heat capacity (constant P), J/kg
C_v : coefficient of heat capacity (constant volume), J/kg
D : molecular diffusion coefficient
E : total energy (internal + kinetic), J/kg
e : specific internal energy, J/kg
$\hat{e}$: unit vector
F : algebraic vector of x-component fluxes
f : species mass fraction
G : algebraic vector of y-component fluxes
h : specific enthalpy, J/kg
k : coefficient of thermal conductivity
L : length of interface in the (x,y)-plane
n : direction cosine
$\hat{n}$: unit normal vector
P : pressure, Pa
Pr : Prandtl number, $(C_p\mu/k)$
Q : algebraic vector of source terms
q : heat flux
R : gas constant, $J/kg/K$
Re : Reynolds number
r : radial coordinate, m
S : control surface of the cell
T : temperature, K
t : time, s
Δt : time step, s
U : algebraic vector of conserved quantities
$\overline{U}$: Riemann invariant
u : x-component of velocity, m/s
v : y-component of velocity, m/s
V : diffusion velocity
ws : wave speed used in the Riemann solver
x : x-coordinate, m
y : y-coordinate, m
z : z-coordinate, m
Z : intermediate variable used in the Riemann solver

α	: weighting function
β	: compression parameter in the MUSCL interpolation
ρ	: density, kg/m^3
γ	: ratio of specific heats
κ	: MUSCL interpolation parameter
λ	: second coefficient of viscosity
τ	: shear stress, Pa
μ	: coefficient of viscosity, $Pa.s$
ψ	: half-angle for the axisymmetric cells, *radians*
θ	: angular coordinate, *radians*
Ω	: cell volume, m^3
Ω'	: volume per radian for the axisymmetric cell
$< \bullet >$	: cell-averaged value
$\sum_{ABCD} \bullet_A \bullet_B$	: $\bullet_A \bullet_B + \bullet_C \bullet_B + \bullet_D \bullet_C + \bullet_D \bullet_A$

Superscripts

n	: time level or iteration level
$*$	: intermediate states for the Riemann solver
$'$	: secondary cell identifier

Subscripts

A, B, C, D	: primary-cell vertices
i, j	: cell-centre indices
is	: species index
$i \pm \frac{1}{2}$	: "vertical" interface
$j \pm \frac{1}{2}$	: "horizontal" interface
MIN	: minimum allowable value
N, S, E, W	: North, South, East, West interface or boundary
n	: normal to interface
t	: tangent to interface
v	: viscous contribution
x,y	: cartesian components
0	: $z = 0$ or $\theta = 0$ plane
L, R	: left state, right state

1 Introduction

In recent years the proliferation of relatively fast computers has popularized the direct calculation of viscous, compressible flows in a time-accurate manner. In some situations, such as the transient hypersonic flow over a model in a shock-tunnel, numerical simulation is the only way to extract detailed information about the flow field. Such computations are very demanding and require computational resources that far exceed those available on a single-processor computer. Hence, flow solvers running on parallel computers are of great interest.

This report describes a code (and set of routines) that integrate the Navier-Stokes equations in a two-dimensional flow domain. Only a single block version of the code using a "tensor product" (or structured) grid is considered here. The multiblock and parallel extensions will be the subjects of future work.

The code is based on the standard cell-centred time-dependent finite-volume formulation as described in [1] - [6]. The governing equations are expressed in integral form over arbitrary quadrilateral cells with the time rate of change of conserved quantities in each cell specified as a summation of the fluxes through the cell interfaces. The fluxes are composed of separate inviscid and viscous components. The inviscid components are computed with a Riemann solver while the viscous fluxes are calculated by application of the divergence theorem.

Section 2 describes the governing equations and programming details for two-dimensional geometries while the governing equations for axisymmetric geometries are given in Appendix A. A set of test cases, designed to exercise various features of the code, is then described in Section 3. These test cases may be used to validate changes made to the code when porting it to different computer architectures.

2 Governing Equations

The integral form of the two-dimensional Navier-Stokes equations in cartesian coordinates can be expressed as

$$\frac{\partial}{\partial t}\int\int_{\Omega} U\, dx\, dy + \int_S (F - F_v)\, dy - \int_S (G - G_v)\, dx = \int\int_{\Omega} Q\, dx\, dy \qquad (1)$$

where

$$U = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \\ \rho f_{is} \end{bmatrix} , \qquad (2)$$

is the algebraic vector of conserved quantities,

$$F = \begin{bmatrix} \rho u \\ \rho u^2 + P \\ \rho v u \\ \rho E u + P u \\ \rho f_{is} u \end{bmatrix} , \quad G = \begin{bmatrix} \rho v \\ \rho u v \\ \rho v^2 + P \\ \rho E v + P v \\ \rho f_{is} v \end{bmatrix} , \qquad (3)$$

are the inviscid flux vectors,

$$F_v = \begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{yx} \\ \tau_{xx} u + \tau_{yx} v + q_x \\ \rho f_{is} V_{x,is} \end{bmatrix} , \quad G_v = \begin{bmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ \tau_{xy} u + \tau_{yy} v + q_y \\ \rho f_{is} V_{y,is} \end{bmatrix} , \qquad (4)$$

are the viscous flux vectors and Q is an algebraic vector of source terms. S is the control surface bounding the control volume Ω. These equations specify the conservation of mass, momentum, energy and the conservation of mass for the individual species in the control volume. They are supplemented by the *equation of state* which relates the pressure to the density and internal energy as

$$P = P(\rho, e) \ . \qquad (5)$$

For a calorically perfect gas, we use

$$P = \rho(\gamma - 1)e \ , \qquad (6)$$

while, for air in chemical equilibrium, we use the curve-fits in [7]. The viscous stresses are given by

$$\tau_{xx} = 2\mu\frac{\partial u}{\partial x} + \lambda\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) \ ,$$

$$\tau_{yy} = 2\mu\frac{\partial v}{\partial y} + \lambda\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) ,$$
$$\tau_{xy} = \tau_{yx} = \mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) , \quad (7)$$

where μ and λ are the first and second coefficients of viscosity. The viscous heat flux is

$$q_x = k\,\partial T/\partial x + \rho\sum h_{is} f_{is} V_{x,is} ,$$
$$q_y = k\,\partial T/\partial y + \rho\sum h_{is} f_{is} V_{y,is} . \quad (8)$$

Currently, the code convects the species without considering their diffusion (i.e. $V_{x,is} = V_{y,is} = 0$). For air, viscosity is evaluated using Sutherland's law

$$\mu = 1.458 \times 10^{-6}\frac{T\sqrt{T}}{T + 110.4} , \quad (9)$$

where T is in degrees Kelvin and μ is in $Pa.s$. Also, Stokes' hypothesis (of zero bulk viscosity) is invoked to give $\lambda = -\frac{2}{3}\mu$ and a constant Prandtl number ($Pr = 0.72$) is used to evaluate the coefficient

$$k = \frac{C_p\mu}{Pr} . \quad (10)$$

For two-dimensional flow without heat sources or chemistry, the source terms in Q are set to zero. For axisymmetric flow, equations (1) - (4) and (7) are replaced by their axisymmetric counterparts discussed in Appendix A.

2.1 Spatial Discretization and Data Storage

The governing equations (1) - (4) are applied to straight-edged quadrilateral cells as shown in Fig. 1. Note that the bounding contour S consists of 4 line segments in the (x, y)-plane and (for 2D geometry) the cell extends 1 unit in the z-direction. A cell-centred discretization is used in which cell-averaged values $< U >$ are associated with the "primary" cell centres. The vertices are labelled $A - D$ and the line integrals in equation (1) (which are taken in the usual counter-clockwise direction) are approximated using the midpoint rule Thus, the semi-discrete equations can be expressed as

$$\frac{d < U >}{dt} + \frac{1}{\Omega}\sum_{ABCD}(F - F_v)(y_B - y_A) - \frac{1}{\Omega}\sum_{ABCD}(G - G_v)(x_B - x_A) = < Q > , \quad (11)$$

where the summations are over the 4 sides of the cell. Given the current flow state and a procedure for computing the average fluxes at the midpoints of the interfaces, equation (11) may the integrated in time as an initial value problem.

The flow domain is divided into a number of blocks with the data for each block stored in a single data structure as defined in Appendix C. Currently, the flow domain consists of a single block. The data structure (labelled A, say) includes both geometry and flow data and flags for boundary conditions.

The structured grid is specified as a two-dimensional array of primary cells with i, j indices ranging from i_{min} to i_{max} and j_{min} to j_{max} respectively. These cells are also called "active" cells (as opposed to "ghost" cells which are used to implement boundary conditions). Using the C language syntax, the density in cell i, j is accessed as A.$Ctr[i][j].rho$. Here A.Ctr is a pointer to an array of pointers, each of which points to a one-dimensional array. See Fig. 2 for a schematic of the data storage. Nonzero values of i_{min} and j_{min} allow storage of ghost cells without the use negative indices or index translation. For convenience, the cell interfaces are labelled "North", "South", "East" and "West". The domain boundaries are labelled "North", "South", "East" and "West" also, and are adjacent to cells with $j = j_{max}$, $j = j_{min}$, $i = i_{max}$ and $i = i_{min}$ respectively.

Referring to Fig. 1, the geometry of the cell i, j is defined by its vertices which are specified as (x, y) coordinate pairs. The vertices A, B, C, D are indexed as A.$Vtx[i][j-1]$, A.$Vtx[i][j]$, A.$Vtx[i-1][j]$, and A.$Vtx[i-1][j-1]$ respectively. The North and South ("horizontal") interfaces are indexed as A.$HF[i][j]$ and A.$HF[i][j-1]$ while the West and East ("vertical") interfaces are indexed as A.$VF[i-1][j]$ and A.$VF[i][j]$ respectively.

The cell volume is computed (via an application of the divergence theorem) as

$$\Omega = \int_{S_{i,j}} x \, dy, \tag{12}$$

where $S_{i,j}$ is the bounding contour for the cell in the (x, y)-plane. This expression is discretized as

$$\Omega_{i,j} \simeq \frac{1}{2} \sum_{ABCD} (x_B + x_A)(y_B - y_A) \ . \tag{13}$$

Note that unit depth has been assumed for the z-direction and the area of the cell in the (x, y)-plane $A_{i,j} = \Omega_{i,j}$. Cell averages of both the primary variables (ρ, u, v, e, p, T) and the conserved variables $(\rho, \rho u, \rho v, \rho E)$ are associated with the cell "centre" or "centroid". The coordinates of the cell centre given by

$$\begin{aligned} x_{Ctr} &= \frac{1}{A_{i,j}} \int\int_{\Omega} x dx dy \ , \\ y_{Ctr} &= \frac{1}{A_{i,j}} \int\int_{\Omega} y dx dy \ , \end{aligned} \tag{14}$$

which are approximated as

$$x_{Ctr} = \frac{1}{8}\sum_{ABCD}(x_B + x_A)^2(y_B - y_A) \ /A_{i,j} \ ,$$
$$y_{Ctr} = \frac{1}{4}\sum_{ABCD}(y_B + y_A)(x_B + x_A)(y_B - y_A) \ /A_{i,j} \ , \qquad (15)$$

after application of the divergence theorem.

An array of "secondary" cells is also defined (see Fig. 3) using the primary cell centres as the new vertices. These secondary cells are used in the calculation of the spatial derivatives required by the viscous stresses (7).

2.2 Inviscid Flux Calculation

The purpose of the inviscid flux routine is to provide estimates for the components of F and G in equation (3) at each cell interface for each time step. This is achieved by first interpolating the flow state (consisting of a set of values for ρ, u, v, e, P, a) to either side of each interface at the start of the time step and then applying a Riemann solver to estimate the flow state at the interface during the time step.

2.2.1 Inviscid Boundary Conditions

Before interpolation, the inviscid boundary conditions are applied by setting up two layers of ghost cells along each of the boundaries. For a supersonic inflow boundary, all of the ghost-cell quantities are specified as fixed while, for a supersonic outflow boundary, the ghost-cell quantities are extrapolated from active cells just inside the boundary. Solid-wall (i.e. tangency) boundary conditions are applied by setting all of the scalar quantities in the ghost cells equal to those in the active cells adjacent to the boundary but setting the ghost-cell velocities to the mirror image of those in the active cells. Note that, for no-slip walls, we apply just the tangency condition at this stage of the calculation.

2.2.2 Interpolation of the Interface State

The state of the flow either side of each interface is interpolated (or reconstructed) from the set of cell averaged states by assuming a variation of the variables within cells. This interpolation is performed independently in each index direction and separately for each

primary variable. For example, the density either side of the vertical interface (i,j) is obtained by a generalized MUSCL interpolation/reconstruction [8] using the expressions

$$\begin{aligned} \rho_L &= \rho_{i,j} + \frac{1}{4}\left[(1-\kappa)\overline{(\Delta-)}_{i,j} + (1+\kappa)\overline{(\Delta+)}_{i,j}\right] , \\ \rho_R &= \rho_{i+1,j} - \frac{1}{4}\left[(1+\kappa)\overline{(\Delta-)}_{i+1,j} + (1-\kappa)\overline{(\Delta+)}_{i+1,j}\right] , \end{aligned} \tag{16}$$

where

$$\begin{aligned} \overline{(\Delta-)}_{i,j} &= MINMOD(\Delta_{i,j},\ \beta\Delta_{i+1,j}) , \\ \overline{(\Delta+)}_{i,j} &= MINMOD(\beta\Delta_{i,j},\ \Delta_{i+1,j}) , \end{aligned} \tag{17}$$

and

$$\Delta_{i,j} = \rho_{i,j} - \rho_{i-1,j} . \tag{18}$$

Interpolation for the other variables and for the horizontal interfaces is done similarly.

Setting the parameter $\kappa = 1/3$ gives an upwind-biased third-order interpolation scheme while setting $\kappa = 1/2$ gives second-order upwind interpolation. The "compression" parameter [9] is restricted to

$$1 \le \beta \le \frac{3-\kappa}{1-\kappa} . \tag{19}$$

We have used $\beta = 2$ for the test cases reported in Section 3. The MINMOD limiter function returns the argument with the minimum magnitude if both arguments have the same sign and returns zero otherwise (see e.g. [10]). To make the code more robust, we impose the conditions $\rho_L, \rho_R \ge \rho_{MIN}$ and $e_L, e_R \ge e_{MIN}$ after interpolation, but before the application of the Riemann solver.

After the interpolation process, a Riemann solver is applied to compute the inviscid flux vectors (3). Note that the solver is applied in a locally rotated frame of reference in which the u-velocity is normal to the interface and the increasing i or j index being on the right side of the interface. The transformation is

$$\begin{aligned} u_{normal} &= +u\, n_x + v\, n_y , \\ v_{tangent} &= -u\, n_y + v\, n_x , \end{aligned} \tag{20}$$

where n_x and n_y are the x- and y-direction cosines of the interface normal.

2.2.3 Riemann Solver

There are a number of Riemann solvers that can be used including "exact" iterative schemes [11] and approximate (noniterative) schemes [12] [13] [14]. The approximate schemes are generally less computationally expensive that the iterative schemes and, because the Riemann solver comsumes a large fraction of the total computational effort, an approximate scheme is favoured. Although the Roe-type solver is popular because it is relatively fast, there are situations (such as the double-Mach-reflection case discussed in Section 3.3) where it occasionally produces spurious results (see [15, 16, 17]). On the other hand, the Osher-type solver [13] is considered to be fairly robust and free of adjustable parameters [18].

We have opted to use a 3-stage approximate solver in which the first stage computes the intermediate pressure and velocity assuming isentropic wave interaction. A second stage, based on the strong-shock relations, may be invoked to improve the first-stage estimate if the pressure jumps across either wave are sufficiently large. In practice, this modification has been required only in extreme conditions such as those found in the bluff-body test case (Section 3.7). The final stage is to select/interpolate the interface state (ρ, u, v, e, P, etc) from the set of left, right and intermediate states. If stage 2 (strong shock modification) is not invoked, the solver is much like Osher's approximate Riemann solver [13].

STAGE 1: The first stage of the Riemann solver assumes that a spatially constant left state (subscript L) and right state (subscript R) interact through a pair of finite-amplitude (and isentropic) compression or rarefaction waves. Perfect gas relations ([19] cited in [11]) are used to obtain the intermediate states (L^*, R^*) in the gas after the passage of left-moving and right-moving waves, respectively. The expressions implemented in the code are

$$P_L^* = P_R^* = P^* = P_L \left[\frac{(\gamma-1)(\overline{U}_L - \overline{U}_R)}{2a_L(1+Z)} \right]^{2\gamma/(\gamma-1)} , \tag{21}$$

and

$$u_L^* = u_R^* = u^* = \frac{\overline{U}_L Z + \overline{U}_R}{1+Z} , \tag{22}$$

where the Riemann invariants are

$$\begin{aligned} \overline{U}_L &= u_L + \frac{2a_L}{\gamma-1} , \text{ and} \\ \overline{U}_R &= u_R - \frac{2a_R}{\gamma-1} , \end{aligned} \tag{23}$$

and the intermediate variable Z is given by

$$Z = \frac{a_R}{a_L} \left(\frac{P_L}{P_R} \right)^{(\gamma-1)/(2\gamma)} . \tag{24}$$

Note that these expressions involve the power operator which is computationally expensive. For a limited range of base and exponent, the standard power function is replaced by the approximate expansion given in Appendix B. In the exceptional situation of $(\overline{U}_L - \overline{U}_R) < 0$, we assume that a (near) vacuum has formed at the cell interface and set all of the interface quantities to minimum values.

STAGE 2: If the pressure jump across either wave is large (say, a factor of 10), then the guess for the intermediate pressure is modified using the strong shock relations.

If $P^* > 10\ P_L$ and $P^* > 10\ P_R$ then both waves are taken to be strong shock waves and the intermediate pressure and velocity can be determined directly as

$$P^* = \frac{\gamma+1}{2}\rho_L\left[\frac{\sqrt{\rho_R}}{\sqrt{\rho_R}+\sqrt{\rho_L}}(u_L - u_R)\right]^2 \quad , \tag{25}$$

and

$$u^* = \frac{\sqrt{\rho_L}\ u_L + \sqrt{\rho_R}\ u_R}{\sqrt{\rho_R}+\sqrt{\rho_L}} \quad . \tag{26}$$

If P^* is greater than P_L or P_R (but not both), the stage-1 estimate for P^* can be improved with two Newton-Raphson steps of the form

$$P^*_{n+1} = P^*_n - F_n\left(\frac{dF_n}{dP^*}\right)^{-1} \quad , \tag{27}$$

where

$$F_n = u^*_L(P^*_n) - u^*_R(P^*_n) \quad , \tag{28}$$

and

$$u^*_L = \begin{cases} \overline{U}_L - \frac{2a_L}{\gamma-1}\left(\frac{P^*}{P_L}\right)^{\frac{\gamma-1}{2\gamma}} \quad , & P^* \le 10\ P_L \quad , \\ u_L - \left(\frac{2P^*}{\rho_L(\gamma+1)}\right)^{1/2} \quad , & P^* > 10\ P_L \quad , \end{cases} \tag{29}$$

$$u^*_R = \begin{cases} \overline{U}_R + \frac{2a_R}{\gamma-1}\left(\frac{P^*}{P_R}\right)^{\frac{\gamma-1}{2\gamma}} \quad , & P^* \le 10\ P_R \quad , \\ u_R + \left(\frac{2P^*}{\rho_R(\gamma+1)}\right)^{1/2} \quad , & P^* > 10\ P_R \quad . \end{cases} \tag{30}$$

During the update, we ensure that $P^* \ge P_{MIN}$ where P_{MIN} is some small value. After updating P^*, the intermediate velocity is evaluated using the relevant strong-shock relation from (29) or (30).

STAGE 3: Now that we have computed the pressure and velocity in the intermediate regions behind the waves, the other intermediate flow properties may be evaluated. Then, the interface conditions used in the inviscid flux vector (3) may be selected or interpolated

from the 4 flow states using the logic shown in Fig. 4. Note that, although only the left-moving wave is discussed below, a similar procedure is used to obtain the flow state behind the right-moving wave.

If the pressure rises across the left-moving wave (i.e. $P^* > P_L$), the left wave is assumed to be a shock and density is obtained from the Rankine-Hugoniot relation as

$$\rho_L^* = \rho_L \left[\frac{(\gamma+1)P^* + (\gamma-1)P_L}{(\gamma+1)P_L + (\gamma-1)P^*} \right] . \tag{31}$$

The specific internal energy is obtained from the equation of state as

$$e_L^* = \frac{P^*}{(\gamma-1)\rho_L^*} , \tag{32}$$

and estimates for the local speed of sound (for later use in the interpolation of the interface properties) are

$$a_L^* = \sqrt{\gamma(\gamma-1)e_L^*} . \tag{33}$$

The velocity of the wave (relative to the initial left state) is given by

$$u_L - ws_L = \left[\frac{\gamma+1}{2} \frac{P_L}{\rho_L} \left(\frac{P^*}{P_L} + \frac{\gamma-1}{\gamma+1} \right) \right]^{1/2} , \tag{34}$$

where ws_L is the velocity of the wave relative to the cell boundaries.

If the pressure falls across the left-moving wave (i.e. $P^* \leq P_L$), the isentropic-wave relations are used to obtain the intermediate properties. The local speed of sound is obtained from the Riemann invariant as

$$a_L^* = (\overline{U}_L - u_L^*)(\gamma-1)/2 , \tag{35}$$

while the specific internal energy is obtained from the sound-speed relation as

$$e_L^* = \frac{(a_L^*)^2}{(\gamma-1)\gamma} . \tag{36}$$

The density is obtained from the equation of state as

$$\rho_L^* = \frac{P^*}{(\gamma-1)e_L^*} . \tag{37}$$

The velocity of the leading-edge of the wave (relative to the initial left state) is given by

$$u_L - ws_L = a_L . \tag{38}$$

In the preceding perfect-gas equations an effective γ may used to include variable gas properties in an approximate manner. We evaluate this effective γ as a density weighted function [20] using

$$\gamma = \alpha\gamma_L + (1-\alpha)\gamma_R\,, \tag{39}$$

where

$$\alpha = \frac{\sqrt{\rho_L}}{\sqrt{\rho_L}+\sqrt{\rho_R}}, \tag{40}$$

and

$$\begin{aligned} \gamma_L &= \frac{P_L}{\rho_L e_L} + 1\,, \\ \gamma_R &= \frac{P_R}{\rho_R e_R} + 1\,. \end{aligned} \tag{41}$$

2.3 Viscous Flux Calculation

After computing the inviscid flux vectors (3), we save the values of u, v and T at the midpoints of the primary-cell interfaces for use in the calculation of the molecular transport of momentum and energy across the interfaces. These values may be either those computed by the Riemann solver or averages of the interpolated left- and right-states.

The spatial derivatives required in the viscous stress terms (7) are obtained by applying the divergence theorem to each of the secondary cells. This gives an average value of the derivative which is then assigned to the primary-cell vertex at the "centre" of the secondary cell. Thus, for the spatial derivatives of temperature at vertex i,j, we compute

$$\begin{aligned} \left(\frac{dT}{dx}\right)_{i,j} &= \frac{1}{2}\sum_{A'B'C'D'}(T_{A'}+T_{B'})(y_{B'}-y_{A'}) \;/\; A_{A'B'C'D'}\;, \\ \left(\frac{dT}{dy}\right)_{i,j} &= \frac{-1}{2}\sum_{A'B'C'D'}(T_{A'}+T_{B'})(x_{B'}-x_{A'}) \;/\; A_{A'B'C'D'}\;, \end{aligned} \tag{42}$$

where

$$A_{A'B'C'D'} = \frac{1}{2}\sum_{A'B'C'D'}(x_{A'}+x_{B'})(y_{B'}-y_{A'})\;, \tag{43}$$

and A', B', C' and D' are the primary-cell centres surrounding the vertex. Spatial derivatives of the u- and v-velocity components are calculated similarly. The viscous fluxes (4) at the interface midpoints are then computed using averages of the viscous stresses at the nearby cell vertices.

2.3.1 Viscous Boundary Conditions

For the velocity field, application of the "no-slip" boundary condition is simply a matter of setting both of the components to zero at the boundary interfaces. The temperature can be set to a specified value for a fixed-T wall or it can be set equal to the value at the adjacent cell centre for an adiabatic wall. For supersonic inflow, supersonic outflow, or a tangency boundary, we leave the values as computed by the inviscid flux routine.

Note that, along the boundaries, the secondary cells are constructed using the two adjacent primary-cell centres and two interface points. The arrangement for each boundary is shown in Fig. 5.

2.4 Time Stepping

The conserved quantities are advanced from time level n to time level $n+1$ with the predictor-corrector scheme

$$\begin{aligned} \Delta U^{(1)} &= \Delta t \frac{dU^{(n)}}{dt} \ , \\ U^{(1)} &= U^{(n)} + \Delta U^{(1)} \ , \\ \Delta U^{(2)} &= \Delta t \frac{dU^{(1)}}{dt} \ , \\ U^{(n+1)} &= U^{(1)} + \frac{1}{2}\left(\Delta U^{(2)} - \Delta U^{(1)}\right) \ , \end{aligned} \tag{44}$$

where the superscripts (1) and (2) indicate intermediate results and the temporal derivative $(\frac{dU}{dt})$ is obtained from equation (11). If a first-order scheme is desired, only the first stage is used and $U^{(n+1)} = U^{(1)}$. Although first-order time-stepping requires fewer operations than second-order time-stepping, it is also less robust.

To maintain stability, the time step is restricted to

$$\Delta t \leq \Delta t_{allowed} = CFL\left(\frac{1}{\Delta t_N} + \frac{1}{\Delta t_E} + \frac{1}{\Delta t_{Viscous}}\right)^{-1} \ , \tag{45}$$

where $\Delta t_{allowed}$ is the smallest value for all cells and

$$CFL \leq \frac{4}{5 - \kappa + \beta(1+\kappa)} \ . \tag{46}$$

is the specified Courant-Friedrichs-Lewy number. Note that restriction (46) is applicable to the schemes developed by Chakravarthy [9] whereas the present procedure seems to be stable for slightly higher values of CFL in some situations. For each cell, the inviscid signal frequencies along the North and East interfaces are approximated as

$$\frac{1}{\Delta t_N} = \frac{|v_{N,tangent}| + a}{L_N} \ , \text{and} \ \frac{1}{\Delta t_E} = \frac{|v_{E,tangent}| + a}{L_E} \ , \tag{47}$$

while the viscous limit [21] is approximated as

$$\frac{1}{\Delta t_{Viscous}} = \frac{4\mu\gamma}{Pr\ \rho}\left(\frac{1}{L_N^2} + \frac{1}{L_E^2}\right) \quad . \tag{48}$$

2.5 Computational Effort

Currently, the code has not been optimized (although the flux calculation routines do vectorize with the default optimization offered by the Cray Standard-C Compiler). On a single processor of a Cray Y-MP, the calculation takes approximately 30 $-$ 35 μs/cell/predictor-corrector step for an inviscid calculation and approximately 40 μs/cell/predictor-corrector step for a viscous calculation. On a SUN Sparc-2 workstation, the calculation takes approximately 1 ms/cell/predictor-corrector step for a viscous calculation. If only the forward time step is used, these times are halved.

3 Test Cases

3.1 One-dimensional Shock Tube

The first test case is the so-called one-dimensional shock tube problem used by Sod [22]. The domain consists of a mesh of 100×2 cells over the rectangular domain $0 < x < 1.0m$, $0 < y < 0.1m$. Reflecting (i.e. tangency) conditions are applied along all domain boundaries and viscous effects are omitted. The gas is calorically perfect with $\gamma = 1.4$. For $x \leq 0.5m$, the initial state is

$$\rho = 1.0\ kg/m^3,\ \ P = 10^5\ Pa,\ \ u = v = 0,\ \ e = 2.5 \times 10^5\ J/kg/K,$$

while, for $x > 0.5m$, it is

$$\rho = 0.125\ kg/m^3,\ \ P = 10^4\ Pa,\ \ u = v = 0,\ \ e = 2.0 \times 10^5\ J/kg/K.$$

At $t = 0$, the hypothetical diaphragm (separating the two initial states) is removed and the inviscid equations are integrated in time to $t \simeq 0.603 \times 10^{-3}s$ with $CFL \simeq 0.5$. MUSCL interpolation with $\kappa = 1/3$ is used. The resulting flow state for a single row of cells in the x-direction is shown in Fig. 6. Comparison with the exact solution (see e.g. [4]) is reasonably good. The shock is captured in three cells and has the correct speed. However, the contact discontinuity is fairly diffuse and the extreme edge of the expansion fan shows some smearing. There is also a small glitch at the base of the rarefaction ($x \simeq 0.5m$) as seen in [9] (Secion 2.6) and [18]. Setting $CFL = 0.01$ produced no discernible change in the plotted solution. The contact discontinuity and shock were spread over the same number of cells and the small glitch was still evident at the base of the rarefaction.

Of the test cases discussed here, this test case is simplest and requires the least memory and processing time. On a Sparc-2 workstation, the total processing time is approximately 27 seconds for 99 time steps (1.4 ms/cell/predictor-corrector time step). Note that this is an overall time and includes file I/O and initialization of the geometry data.

3.2 High-Temperature Shock Tube

This case is similar to Sod's shock tube problem but is more demanding as it has pressure and temperature jumps that are large enough for thermodynamic and chemical effects to be significant.

The flow domain covers $0 < x < 1.0m$, $0 < y < 0.1m$ and is divided into a mesh of 200×2 rectangular cells. The initial flow state is the same as that used by Grossman and Walters [23]. For $x < 0.5m$, we set

$$\rho = 2.641\ kg/m^3,\quad P = 10.09 \times 10^6\ Pa,\quad u = v = 0,\quad e = 21.82 \times 10^6\ J/kg/K,$$

while, for $x > 0.5m$, we set

$$\rho = 1.174\ kg/m^3,\quad P = 0.1006 \times 10^6\ Pa,\quad u = v = 0,\quad e = 12.48 \times 10^6\ J/kg/K.$$

The gas is now assumed to be air in chemical equilibrium with the pressure, temperature and local sound speed specified as curve fits on density and internal energy [7].

Again, the hypothetical diaphragm is removed at $t = 0$ and the governing equations are integrated forward in time with $CFL = 0.5$ and $\kappa = 1/3$. Figure 7 shows the flow state at $t = 0.125 \times 10^{-3}s$. Although an exact solution was not included in this figure, the finite-volume solution appears to be in reasonable agreement with the results published in [23]. This agreement indicates that the use of an effective γ to approximate variable gas properties in the Riemann solver is a reasonable approach for this type of problem.

On a Sparc-2 workstation, this case requires approximately 250 seconds of cpu time for 191 time steps (3.2 ms/cell/predictor-corrector time step).

3.3 Double Mach Reflection

To test the code's ability to capture multidimensional dicontinuities, we examine the double Mach reflection case 10 from Glaz, *et al* [24].

Figure 8 shows the flow domain which is divided into 200×100 cells. These cells are equally spaced in the x-direction and their vertical interfaces are aligned with the y-axis. For each x-station, the cells are equally spaced in the y-direction.

The gas is a calorically perfect gas with $\gamma = 1.4$ and has initial conditions of

$$\rho = 6.82 \times 10^{-2}\ kg/m^3,\quad P = 6.0 \times 10^3\ Pa,\quad u = v = 0,$$

$$e = 2.183 \times 10^5\ J/kg/K,$$

throughout the domain. At $t = 0$, a constant supersonic inflow with

$$\rho = 0.3028\ kg/m^3,\quad P = 95.88 \times 10^3\ Pa,\quad u = 1006m/s,\quad v = 0,$$

$$e = 7.913 \times 10^5\ J/kg/K,\quad M = 1.51,$$

is applied to the West ($x = 0$) boundary. Tangency conditions are applied along the North ($y = 1.0m$) and South boundaries and zero-order extrapolation is used at the East ($x = 1.0m$) boundary. The governing equations for inviscid flow are integrated forward in time with $CFL = 0.5$ and $\kappa = 1/3$.

Initially a planar shock is established and propagated into the flow domain with a shock Mach number of 3.72. On encountering the ramp, this "primary" shock is reflected and a (nearly) self-similar flow is established. Figure 9 shows the density contours at two times after the shock has encountered the ramp. The main features of the flow are the (1) primary shock (still travelling from left to right), (2) a detached (curved) shock forming over the leading edge of the ramp, (3) a Mach stem from the primary shock to the ramp, (4) another Mach stem from the primary shock to the detached shock, and (5) a pair shear layers propagating from the the intersections of the shocks. At later times, the detached shock continues to propagate upstream and the flow field becomes subsonic. Figure 10 shows a comparison of the present solution at $t = 0.7ms$ and an interferogram from [24]. Agreement is reasonable given the uncertainty in the physical gas properties and the grid resolution used in the calculation. Note that there is no evidence of a distorted Mach stem near the wall as seen in some calculations made with the Roe-type approximate Riemann solver [15, 16, 17].

3.4 Flat Plate Boundary Layer

The implementation of the two-dimensional viscous terms was validated by computing 2 cases of a supersonic, laminar boundary layer over a flat plate.

The flow geometry (for both cases) consists of a flat plate, $1.0m$ long, aligned with a uniform Mach 2 flow. The gas is considered calorically perfect with $\gamma = 1.4$, $R = 287J/kg/K$ and a constant Prandtl number of 0.72. The computational domain, as shown in Fig. 11, is shaped to include the leading-edge interaction shock (LEIS). The domain is divided into $N \times N$ cells which are clustered toward the plate surface and toward the inflow boundary using one of Robert's stretching transformations [25] (see also Section 5-6.1 in [26]).

For case 1, we set $N = 100$ and apply supersonic free-stream conditions of

$$\rho = 0.0404\ kg/m^3, \quad u = 597.3\ m/s, \quad v = 0, \quad e = 1.592 \times 10^5\ J/kg,$$

to the West and North boundaries. These conditions correspond to

$$M = 2.0, \quad Re_L = 1.65 \times 10^6, \quad T = 222\ K,$$

as used in [27]. The South boundary is set to be a no-slip wall with a fixed temperature $T_{wall} = 222K$ while the East boundary conditions are obtained by zero-order (or constant)

extrapolation. Initially, the flow throughout the domain is set at free-stream conditions and the governing equations are integrated forward in time using first-order (Euler) time-stepping with $CFL = 0.8$ and $\kappa = 1/3$. Figure 12 shows the pressure field at $t = 7.0ms$. The only apparent feature is the weak shock propagating into the flow from the leading edge of the plate. However, a boundary layer develops along the plate and attains a total thickness of approximately $0.005m$ by the end of the plate. Figure 13 compares the temperature and x-velocity profiles at $x = 0.941m$ with profiles computed by a spectrally-accurate boundary layer code [28]. The shear stress estimates agree to within 3% at this time and the flow is still approaching steady state (slowly). For this x-station, the first cell-centre off the wall has $y^+ \simeq 5$ where

$$y^+ = \frac{y\rho_w\sqrt{\tau_w/\rho_w}}{\mu_w} \quad . \tag{49}$$

Case 2 has the same flow geometry but has

$$N = 50, \quad \rho = 0.00404\ kg/m^3, \quad Re_L = 1.65 \times 10^5.$$

All other parameters are the same as case 1. Figure 14 shows the cell-centre mesh and the pressure field at $t = 8.0ms$. The LEIS is now much stronger and the boundary layer, which scales with $\sqrt{Re}$, is approximately 3 times thicker. Fig. 15 shows the boundary layer profiles at $x = 0.916m$ where the displacement thickness is $5.10mm$. Again, there is reasonable agreement with the spectrally accurate boundary layer solution. This indicates that the weak leading-edge shock influences the boundary layer very little. The processing time required for this case is approximately 1.6 hours on a single processor of a Cray Y-MP for approximately 116000 time steps (20 μs/cell/Euler time step).

3.5 Inviscid Flow over a Cone

Inviscid flow over a cone is used to test the axisymmetric formulation of Appendix A. In the steady-state limit, the shock (and other constant property lines) are straight and there is an "exact" solution [29] (see also Ch. 10, [30]). Figure 16 shows the flow geometry for a 20° half-angle cone whose axis of symmetry is along the x-axis. The flow domain is divided into a mesh of 100×100 cells. The cells are equally spaced in the x-direction and their vertical interfaces are aligned with the y-axis. For each x-station, the cells are equally spaced in the y-direction.

Although we are interested in the quality of the steady-state solution for this test, we simulate the flow using the same initial and inflow conditions as the double Mach reflection

case (Section 3.3). The gas is a calorically perfect with $\gamma = 1.4$ and initial conditions throughout the domain are

$$\rho = 6.82 \times 10^{-2}\ kg/m^3, \quad P = 6.0 \times 10^3\ Pa, \quad u = v = 0,$$

$$e = 2.183 \times 10^5\ J/kg/K \quad .$$

At $t = 0$, a constant supersonic inflow with

$$\rho = 0.3028\ kg/m^3, \quad P = 95.88 \times 10^3\ Pa, \quad u = 1006 m/s, \quad v = 0,$$

$$e = 7.913 \times 10^5\ J/kg/K, \quad M = 1.51,$$

is applied to the West boundary. Tangency conditions are applied along the North and South boundaries and zero-order extrapolation is used at the East boundary. The governing equations for inviscid flow are integrated forward in time with $CFL = 0.5$ and high-order MUSCL interpolation.

Figure 17 shows the density field at two instants after $t = 0$. By $t = 1.0ms$, the primary (normal) shock has left the flow domain and conical flow is being established over the nose of the cone. Figure 18 shows both the pressure and density fields (at $t = 5.0ms$) when the flow has nearly reached steady state. Except for small deviations, the contours are straight lines aligned with the (conical) generators propagating from the cone vertex. The shock angle closely matches the value of 49° taken from Chart 5 in [31]. Note that, for an equivalent two-dimensional situation, there is no "attached-shock" solution.

On a Sparc-2 workstation, this case requires 5.36 hours cpu time to take 2410 time steps (0.8 ms/cell/predictor-corrector time step).

3.6 Viscous Flow along a Cylinder

The implementation of the axisymmetric viscous terms is examined by computing the supersonic, laminar boundary layer along a hollow cylinder. Except for the axisymmetric geometry, this case is similar to case 2 of the flat-plate boundary layer in Section 3.4.

The flow geometry consists of a hollow cylinder aligned with the x axis. The cylinder is $1.0m$ long and has a $0.005m$ radius. The free stream is a uniform supersonic flow with $M = 2$. The gas is considered calorically perfect with $\gamma = 1.4$, $R = 287 J/kg/K$ and a constant Prandtl number of 0.72. The computational domain, as shown in Fig. 19, is shaped to include the leading-edge interaction shock (LEIS) and is divided into 50×50 cells (as done for the flat-plate boundary layer).

We apply supersonic free-stream conditions of

$$\rho = 0.00404\ kg/m^3,\quad u = 597.3\ m/s,\quad v = 0,\quad e = 1.592 \times 10^5\ J/kg,$$

$$M = 2.0,\quad Re_L = 1.65 \times 10^5,\quad T = 222\ K,$$

to the West and North boundaries. The South boundary is set to be a no-slip wall with a fixed temperature $T_{wall} = 222K$ while the East boundary conditions are obtained by zero-order (or constant) extrapolation. Initially, the flow throughout the domain is set at free-stream conditions and the governing equations are integrated forward in time using first-order (Euler) time-stepping with $CFL = 0.8$ and high-order MUSCL interpolation. Figure 20 shows the pressure field at $t = 8.0ms$. Compared to the two-dimensional situation, the LEiS is much weaker and the boundary layer at the end of the cylinder is thinner. Figure 21 compares the temperature and x-velocity profiles at $x = 0.916m$ for both a 50×50 grid and a 70×70 grid with profiles computed by a spectrally-accurate boundary layer code [28]. Here, the displacement thickness is $4.22mm$ which is approximately 20% less than the corresponding value on the flat plate. Although agreement is generally good for both grids, there is noticeable improvement in the temperature profile for the finer grid.

3.7 High Mach Number Flow around a Sphere.

The robustness of the code is demonstrated by computing a Mach 12 flow over a sphere. This case is difficult because the shock in front of the sphere is very strong and because there is a geometric singularity along the stagnation line.

Figure 22 shows the flow geometry and the 60×60 mesh of cell centres. The South boundary is the $y = 0$ symmetry line (and stagnation line) while the East boundary is the surface of the sphere. A tangency condition is applied at this surface. For case 1, free-stream conditions of

$$\rho = 0.5097\ kg/m^3,\quad P = 42717\ Pa,\quad e = 2.095 \times 10^5\ J/kg,$$

$$u = 4120.6(1 - \exp(-5 \times 10^{-6}t))\ m/s,\quad v = 0,\quad M_{nominal} = 12,$$

are applied to the West boundary. Flow conditions at the North boundary are obtained by zero-order (constant) extrapolation. The shape of the West boundary is derived from the shock-position correlations in [32]. Initial conditions throughout the domain are set to

$$\rho = 0.5097\ kg/m^3,\quad P = 42717\ Pa,\quad e = 2.095 \times 10^5\ J/kg,\quad u = 0,\quad v = 0,$$

The inviscid equations are then integrated forward in time using first-order interpolation, Euler time-stepping and $CFL = 0.5$. Figure 23 shows the flow field (density contours) at an early time and again after the flow has approached steady state. The shock wave moves off the body and approaches its steady-state position in a well behaved manner. For this case, a finite-difference scheme using Roe-type flux-difference splitting required a rather large value for its entropy-fix parameter in order to obtain a physically reasonable solution (J. White, NASA Langley Research Centre, private communication). Although the shock shape is well behaved for the present code, there are significant small-scale disturbances just behind the shock, in the subsonic region. These disturbances appear to caused by small perturbations to the shock position and have the scale of the local mesh spacing. Discrete points from experimentally derived correlations [32] are plotted on the later-time solution. On the whole, agreement is good. The largest deviations are near the outflow boundary where the flow field may be still developing.

Two other calculations are included to show the effect of viscosity on the small-scale disturbances in the subsonic region. To make the viscous effects larger, the free-stream density and pressure are lowered to

$$\rho = 0.5097 \times 10^{-2}\ kg/m^3, \quad P = 427.1 Pa\ .$$

The governing equations are integrated in time with the viscous terms included but the East boundary condition is still a tangency condition. For case 2, the first-order interpolation is used and the result is shown in Fig. 24(a). The extra dissipation has damped the disturbances in the subsonic region. High-order MUSCL interpolation is used for case 3 and as shown in Fig. 24(b), the result is essentially the same except for some slightly noisier contours away from the axis. Profiles of density and pressure along the line of cells adjacent to the x-axis are shown in Fig. 25. The shock appears to be captured in 2 or 3 cells with no oscillation and the density jump is close to the ideal strong-shock value of 6 (see e.g. [33] Section 2.2). The pressure ratio from free-stream to the stagnation point is 185.8 which is very close to the ideal value of 185.9 for $M = 12$ (see e.g. [30] Table A.2).

4 Concluding Remarks

This report has described a program for the time-integration of the Navier-Stokes equations on a two-dimensional structured mesh. The program is based on a cell-centred, finite-volume formulation and uses a three-stage Riemann solver to compute the inviscid fluxes. Viscous fluxes are computed by applying the divergence theorem to the flow data on a set of secondary cells. Grid metrics are not required. Time stepping is performed with either an Euler or a predictor-corrector scheme.

Currently, the program is restricted to a single-block, structured grid. However, the code modules are written so that they may be applied to any number of such blocks. All that is required is the addition of a set of routines to exchange boundary data and a modified time-stepping procedure. This extension and the addition of chemical source terms will be the subject of future work.

Acknowledgements

Thanks to Jeff Scroggs (North Carolina State University), Dean Eklund, John Korte, Joe Morrison, David Pruett and Jeff White (NASA Langley Research Centre) for many useful discussions and suggestions.

References

[1] R. C. Swanson and E. Turkel. A multistage time-stepping scheme for the Navier-Stokes equations. ICASE Report 84-62, 1990.

[2] A. Arnone and R. C. Swanson. A Navier-Stokes solver for cascade flows. ICASE Report 88-32, 1988.

[3] C. Hirsch. *Numerical Computation of Internal and External Flows. Volume 1: Fundamentals of Numerical Discretization.* John Wiley & Sons, 1988.

[4] C. Hirsch. *Numerical Computation of Internal and External Flows. Volume 2: Computational Methods for Inviscid and Viscous Flows.* John Wiley & Sons, 1990.

[5] R. W. Walters, D. C. Slack, P. Cinnella, M. Applebaum, and C. Frost. A user's guide to GASP. Technical report, Virginia Polytechnic Institute and State University, 1990.

[6] M. Vinokur. An analysis of finite-difference and finite-volume foundations of conservation laws. *Journal of Computational Physics*, 81(1):1–52, 1989.

[7] S. Srinivasan and J. C. Tannehill. Simplified curve fits for the transport properties of equilibrium air. NASA Contractor Report 178411, Iowa State University, 1987.

[8] W. K. Anderson, J. L. Thomas, and B. van Leer. A comparison of finite volume flux vector splittings for the Euler equations. AIAA Paper 85-0122, 1985.

[9] S. R. Chakravarthy. Development of upwind schemes for the Euler equations. NASA Contractor Report 4043, 1987.

[10] P. K. Sweby. High resolution schemes using flux limiters for hyperbolic conservation laws. *SIAM J. Numer. Anal.*, 21:995–1010, 1984.

[11] J. J. Gottlieb and C. P. T. Groth. Assessment of Riemann solvers for unsteady one-dimensional inviscid flows of perfect gas. *Journal of Computational Physics*, 78(2):437–458, 1988.

[12] P. L. Roe. Approximate Riemann solvers, parameter vectors, and difference schemes. *Journal of Computational Physics*, 43:357–372, 1981.

[13] S. Osher and F. Solomon. Upwind difference schemes for hyperbolic systems of conservation laws. *Mathematics of Computation*, 38(158):339–374, 1982.

[14] J. K. Dukowicz. A general, non-iterative Riemann solver for Godunov's method. *Journal of Computational Physics*, 61(1):119–137, 1985.

[15] J. Casper. Finite-volume application of high-order ENO schemes to two-dimensional boundary-value problems. AIAA Paper 91-0631, 1991.

[16] J. J. Quirk. An adaptive grid algorithm for computational shock hydrodynamics. Ph.D. Thesis, Cranfield Institute of Technology, 1991.

[17] D. F. Hawken and J. J. Gottlieb. Prediction of two-dimensional time-dependent gasdynamic flows for hypersonic studies. UTIAS Report 335, 1990.

[18] R. Abgrall, L. Fezoui, and J. Talandier. An extension of Osher's Riemann solver for chemical and vibrational nonequilibrium gas flows. *Int. J. for Numerical Methods in Fluids*, Submitted, 1991.

[19] S. K. Godunov (Ed). *Numerical Solution of Multidimensional Problems in Gasdynamics.* Nauka, Moscow, 1976.

[20] T. A. Edwards. The effect of exhaust plume/ afterbody interaction on installed scramjet performance. NASA Technical Memorandum 101033, 1988.

[21] R. C. Swanson, E. Turkel, and J. A. White. An effective multigrid method for high-speed flows. In *Fifth Copper Mountain Conference on Multigrid Methods*, 1991.

[22] G. A. Sod. A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws. *Journal of Computational Physics*, 27(1):1–31, 1978.

[23] B. Grossman and R. W. Walters. Analysis of flux-split algorithms for Euler's equations with real gases. *A.I.A.A. Journal*, 27(5):524–531, 1989.

[24] H. M. Glaz, P. Colella, I. I. Glass, and R. L. Deschambault. A detailed numerical, graphical, and experimental study of oblique shock wave reflections. UTIAS Report 285, University of Toronto, 1986.

[25] G. O. Roberts. Computational meshes for boundary layer problems. In *Lecture Notes in Physics, 8*, pages 171–177. Springer-Verlag, 1971.

[26] D. A. Anderson, J. C. Tannehill, and R. H. Pletcher. *Computational Fluid Mechanics and Heat Transfer.* Hemisphere, New York, 1984.

[27] J. J. Korte. An explicit, upwind algorithm for solving the parabolized Navier-Stokes equations. Ph. d. dissertation, North Carolina State University, 1989.

[28] C. D. Pruett and C. L. Street. A spectral collocation method for compressible, nonsimilar boundary layers. *Int. J. Num. Meth. Fluids*, Accepted for publication, 1991.

[29] G. I. Taylor and J. W. Maccoll. The air pressure on a cone moving at high speed. *Proc. Roy. Soc. (London) Ser. A*, 139:278–311, 1933.

[30] J. D. Anderson. *Modern Compressible Flow: with Historical Perspective.* McGraw-Hill, New York, 1982.

[31] Ames Research Staff. Equations, tables and charts for compressible flow. NACA Report 1135, 1953.

[32] F. S. Billig. Shock-wave shapes around spherical- and cylindrical-nosed bodies. *J. Spacecraft and Rockets*, 4(5):882–883, 1967.

[33] J. D. Anderson. *Hypersonic and High Temperature Gas Dynamics.* McGraw-Hill, New York, 1989.

[34] D. R. Lund. Numerical modeling of supersonic turbulent reacting free shear layers. Ph.D. Thesis, North Carolina State University, Department of Mechanical Engineering, 1989.

A Axisymmetric Flow Geometry

We now consider an axisymmetric flow with velocity

$$\begin{aligned} \vec{u} &= u\,\hat{e}_x + v\,\hat{e}_y + w\,\hat{e}_z \ , \\ u &= u_0(x,r) \ , \\ v &= v_0(x,r)\,\cos\theta \ , \\ w &= v_0(x,r)\,\sin\theta \ . \end{aligned} \tag{50}$$

Here $r \equiv y_0$ and θ are the polar coordinates in the (y,z)-plane as shown in Fig. 26. The subscript 0 refers to the $\theta = 0$ (or the $z = 0$) plane. All other primary variables (i.e. P, T, e, f_{is}) are functions of x and r only. Derivatives are related by

$$\begin{aligned} \frac{\partial}{\partial x} &= \frac{\partial}{\partial x} \ , \\ \frac{\partial}{\partial y} &= -\frac{\sin\theta}{r}\frac{\partial}{\partial\theta} + \cos\theta\frac{\partial}{\partial r} \ , \\ \frac{\partial}{\partial z} &= +\frac{\cos\theta}{r}\frac{\partial}{\partial\theta} + \sin\theta\frac{\partial}{\partial r} \ . \end{aligned} \tag{51}$$

Following the approach of Vinokur [6], we introduce an axisymmetric cell of extent 2ψ radians in the (y,z)-plane. Figure 27 which shows an segment of an axi-symmetric volume element with the axis of symmetry aligned with the x-axis. Note the "Front" and "Back" interfaces at angles $\theta = +\psi$ and $\theta = -\psi$ respectively. These interfaces have unit normals given by

$$\begin{aligned} \hat{n}_{\theta=+\psi} &= -\sin\psi\,\hat{e}_y + \cos\psi\,\hat{e}_z \ , \\ \hat{n}_{\theta=-\psi} &= -\sin\psi\,\hat{e}_y - \cos\psi\,\hat{e}_z \ . \end{aligned} \tag{52}$$

Due to the axial-symmetry of the flow field, there is zero normal velocity at these planes.

The cell volume is now

$$\Omega = 2\psi \int_{S_0} yx\,dy \ , \tag{53}$$

where S_0 is the contour of the cell in the (x,y)-plane. Also, we define the modified cell volume (or volume per radian) to be

$$\Omega' = \frac{\Omega}{2\psi} \ . \tag{54}$$

In the code, equation (54) is approximated by

$$\Omega \simeq \sum_{ABCD} \frac{(y_A + y_B)}{2}\frac{(x_A + x_B)}{2}(y_B - y_A) \ . \tag{55}$$

A.1 Governing Equations in Cartesian Coordinates

We now consider the cartesian form of the Navier-Stokes equations in three dimensions and then substitute the axisymmetric expressions into each of the terms. The z-momentum equation will then be dropped from the set and replaced with the approximation that the pressure at the Front and Back interfaces is the same as that at the cell centre. The goal is to reduce the full three-dimensional system to a system of quasi-two-dimensional equations which can be used in place of equations (1) - (4), (7) in Section 2.

The governing equations can be expressed as

$$\frac{\partial}{\partial t}\int_\Omega U\,dx\,dy\,dz + \int_S \left[(F-F_v)\,\hat{e}_x + (G-G_v)\,\hat{e}_y + (H-H_v)\,\hat{e}_z\right]\cdot\hat{n}\,dS = \int_\Omega Q\,dx\,dy \tag{56}$$

where

$$U = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \\ \rho f_{is} \end{bmatrix}, \tag{57}$$

$$F = \begin{bmatrix} \rho u \\ \rho u^2 + P \\ \rho v u \\ \rho w u \\ \rho E u + P u \\ \rho f_{is} u \end{bmatrix}, \quad G = \begin{bmatrix} \rho v \\ \rho u v \\ \rho v^2 + P \\ \rho w v \\ \rho E v + P v \\ \rho f_{is} v \end{bmatrix}, \quad H = \begin{bmatrix} \rho w \\ \rho u w \\ \rho v w \\ \rho w^2 + P \\ \rho E w + P w \\ \rho f_{is} w \end{bmatrix}, \tag{58}$$

$$F_v = \begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{yx} \\ \tau_{zx} \\ \tau_{xx}u + \tau_{yx}v + \tau_{zx}w + q_x \\ \rho f_{is} V_{x,is} \end{bmatrix}, \quad G_v = \begin{bmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ \tau_{zy} \\ \tau_{xy}u + \tau_{yy}v + \tau_{zy}w + q_y \\ \rho f_{is} V_{y,is} \end{bmatrix}, \tag{59}$$

$$H_v = \begin{bmatrix} 0 \\ \tau_{xz} \\ \tau_{yz} \\ \tau_{zz} \\ \tau_{xz}u + \tau_{yz}v + \tau_{zz}w + q_z \\ \rho f_{is} V_{z,is} \end{bmatrix}. \tag{60}$$

The individual viscous stress terms are

$$\tau_{xx} = 2\mu\frac{\partial u}{\partial x} + \lambda\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}\right),$$

$$
\begin{aligned}
\tau_{yy} &= 2\mu\frac{\partial v}{\partial y} + \lambda\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}\right) \quad , \\
\tau_{zz} &= 2\mu\frac{\partial w}{\partial z} + \lambda\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}\right) \quad , \\
\tau_{xy} = \tau_{yx} &= \mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) \quad , \\
\tau_{xz} = \tau_{zx} &= \mu\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right) \quad , \\
\tau_{yz} = \tau_{zy} &= \mu\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right) \quad , \qquad (61)
\end{aligned}
$$

and the heat-flux terms are

$$
\begin{aligned}
q_x &= -k\frac{\partial T}{\partial x} \quad , \\
q_y &= -k\frac{\partial T}{\partial y} \quad , \\
q_z &= -k\frac{\partial T}{\partial z} \quad . \qquad (62)
\end{aligned}
$$

Currently, we set the diffusion velocities V_{is} to zero.

A.2 Application to the Axisymmetric Cell

We now substitute the axisymmetric expressions into the cartesian equations (56) - (62). The algebraic vectors U, F, G and H are now

$$
U = \begin{bmatrix} \rho_0 \\ \rho_0 u_0 \\ \rho_0 v_0 \cos\theta \\ \rho_0 v_0 \sin\theta \\ \rho_0 E_0 \\ \rho_0 f_{is,0} \end{bmatrix} \quad , \qquad (63)
$$

$$
F = \begin{bmatrix} \rho_0 u_0 \\ \rho_0 u_0^2 + P_0 \\ \rho_0 v_0 \cos\theta \; u_0 \\ \rho_0 v_0 \sin\theta \; u_0 \\ \rho_0 E_0 u_0 + P_0 u_0 \\ \rho_0 f_{is_0} u_0 \end{bmatrix} \quad , \quad G = \begin{bmatrix} \rho_0 v_0 \\ \rho_0 u_0 v_0 \cos\theta \\ \rho_0 v_0^2 \cos^2\theta + P_0 \\ \rho_0 v_0 \sin\theta \; v_0 \cos\theta \\ \rho_0 E_0 v_0 \cos\theta + P_0 v_0 \cos\theta \\ \rho_0 f_{is,0} v_0 \cos\theta \end{bmatrix} \quad , \qquad (64)
$$

$$
H = \begin{bmatrix} \rho_0 v_0 \sin\theta \\ \rho_0 u_0 v_0 \sin\theta \\ \rho_0 v_0 \cos\theta \; v_0 \sin\theta \\ \rho_0 v_0^2 \sin^2\theta + P_0 \\ \rho_0 E_0 v_0 \sin\theta + P_0 v_0 \sin\theta \\ \rho_0 f_{is,0} v_0 \sin\theta \end{bmatrix} \quad . \qquad (65)
$$

Taking $\theta << 1$ and dropping some of the high-order terms in the viscous-stress expressions gives

$$\begin{aligned}
\tau_{xx} &= 2\mu\frac{\partial u_0}{\partial x} + \lambda\left(\frac{\partial u_0}{\partial x} + \frac{\partial v_0}{\partial r} + \frac{v_0}{r}\right) \quad , \\
\tau_{yy} &= 2\mu\frac{\partial v_0}{\partial r} + \lambda\left(\frac{\partial u_0}{\partial x} + \frac{\partial v_0}{\partial r} + \frac{v_0}{r}\right) \quad , \\
\tau_{zz} &= 2\mu\frac{v_0}{r} + \lambda\left(\frac{\partial u_0}{\partial x} + \frac{\partial v_0}{\partial r} + \frac{v_0}{r}\right) \quad , \\
\tau_{xy} = \tau_{yx} &= \mu\cos\theta\left(\frac{\partial u_0}{\partial r} + \frac{\partial v_0}{\partial x}\right) \quad , \\
\tau_{xz} = \tau_{zx} &= \mu\sin\theta\left(\frac{\partial u_0}{\partial r} + \frac{\partial v_0}{\partial x}\right) \quad , \\
\tau_{yz} = \tau_{zy} &= 2\mu\sin\theta\cos\theta\left(\frac{\partial v_0}{\partial r} - \frac{v_0}{r}\right) \quad , \qquad (66)
\end{aligned}$$

and

$$\begin{aligned}
q_x &= -k\frac{\partial T_0}{\partial x} \quad , \\
q_y &= -k\cos\theta\frac{\partial T_0}{\partial r} \quad , \\
q_z &= -k\sin\theta\frac{\partial T_0}{\partial r} \quad . \qquad (67)
\end{aligned}$$

Assuming small ψ, we can integrate in θ and drop the z-momentum equation to obtain

$$\begin{aligned}
\frac{\partial < U_0 >}{\partial t}\Omega &+ 2\psi\int_{S_0} r\,(F - F_v)_{\theta=0}\,dr \\
&- 2\psi\int_{S_0} r\,(G - G_v)_{\theta=0}\,dx \\
&+ \int_{S_{Front}} [-\sin\psi\,(G - G_v)_{\theta=+\psi} + \cos\psi\,(H - H_v)_{\theta=+\psi}]\cdot\hat{n}_{\theta=+\psi}\,dS \\
&+ \int_{S_{Back}} [-\sin\psi\,(G - G_v)_{\theta=-\psi} - \cos\psi\,(H - H_v)_{\theta=-\psi}]\cdot\hat{n}_{\theta=-\psi}\,dS \\
&= 0 \quad , \qquad (68)
\end{aligned}$$

where the line integrals over S_0 are analogous to the contributions from the North, South, East and West interfaces in the two-dimensional case while the integrals over S_{Front} and S_{Back} are the contributions from the $\theta = +\psi$ (Front) and $\theta = -\psi$ (Back) interfaces respectively.

A.3 Treatment of the interfaces at $\theta = \pm\psi$

Now, we move the Front and Back interface contributions to the right-hand-side and approximate the integrals by an average times the interface area, A. The result may be considered

an *effective* source term

$$\begin{aligned} Q'\,\Omega \;=\; & -A\left[-\sin\psi(G-G_v)_{\theta=+\psi}+\cos\psi(H-H_v)_{\theta=+\psi}\right] \\ & -A\left[-\sin\psi(G-G_v)_{\theta=-\psi}-\cos\psi(H-H_v)_{\theta=-\psi}\right] \quad , \end{aligned} \tag{69}$$

where

$$A=\int_{S_0} x\; dy \quad . \tag{70}$$

Substituting the expressions for G, G_v, H and H_v, and taking the limit of small ψ, we obtain

$$Q'\,\Omega = 2\psi A \begin{bmatrix} 0 \\ 0 \\ P_0-\tau_{\theta\theta} \\ 0 \\ 0 \end{bmatrix} \quad , \tag{71}$$

where we define

$$\tau_{\theta\theta} \equiv 2\mu\frac{v_0}{r}+\lambda\left(\frac{\partial u_0}{\partial x}+\frac{\partial v_0}{\partial r}+\frac{v_0}{r}\right) \quad . \tag{72}$$

Note that the quantities in Q' are evaluated at the cell centre.

A.4 Summary of the Axisymmetric Equations

Reverting to (x,y) notation without the zero subscripts, and dividing through by equation (54) gives the governing equations for axisymmetric flow as

$$\frac{d<U>}{dt}+\frac{1}{\Omega'}\int_S (yF-yF_v)\;dy-\frac{1}{\Omega'}\int_S (yG-yG_v)\;dx=Q' \tag{73}$$

where the U, F and G vectors are

$$U \;=\; \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \\ \rho f_{is} \end{bmatrix} \quad , \quad yF \;=\; y\begin{bmatrix} \rho u \\ \rho u^2+P \\ \rho vu \\ \rho Eu+Pu \\ \rho f_{is}u \end{bmatrix} \quad , \quad yG \;=\; y\begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2+P \\ \rho Ev+Pv \\ \rho f_{is}v \end{bmatrix} \quad . \tag{74}$$

Note that, except for the "y" premultiplying factor, they are the same as those in the planar two-dimensional situation defined in (3). The viscous terms are

$$yF_v = \begin{bmatrix} 0 \\ y\tau_{xx} \\ y\tau_{xy} \\ y\tau_{xx}u+y\tau_{xy}v+yq_x \\ y\rho V_{x,is}Y_{is} \end{bmatrix} \quad , \quad yG_v = \begin{bmatrix} 0 \\ y\tau_{yx} \\ y\tau_{yy} \\ y\tau_{yx}u+y\tau_{yy}v+yq_y \\ y\rho V_{y,is}Y_{is} \end{bmatrix} \quad , \tag{75}$$

where

$$\begin{aligned} \tau_{xx} &= 2\mu\frac{\partial u}{\partial x} + \lambda\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{v}{y}\right) \quad , \\ \tau_{yy} &= 2\mu\frac{\partial v}{\partial y} + \lambda\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{v}{y}\right) \quad , \\ \tau_{xy} = \tau_{yx} &= \mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) \quad , \end{aligned} \tag{76}$$

and

$$\begin{aligned} q_x &= -k\frac{\partial T}{\partial x} \quad , \\ q_y &= -k\frac{\partial T}{\partial y} \quad . \end{aligned} \tag{77}$$

Treating the viscous contributions in the form $y\tau$ avoids any difficulties with the geometry singularity at $y = 0$. The effective source term (containing the Front- and Back- interface contributions) is

$$Q' = \begin{bmatrix} 0 \\ 0 \\ (P - \tau_{\theta\theta})A/\Omega' \\ 0 \\ 0 \end{bmatrix} \quad . \tag{78}$$

where

$$\tau_{\theta\theta} = 2\mu\frac{v}{y} + \lambda\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{v}{y}\right) \quad . \tag{79}$$

These equations are equivalent to those used by Eklund [34].

Finally, we note that the treatment described here (including discretization) preserves free-stream properties. This was checked by running the cylinder test case in Section 3.6 with tangency conditions along the cylinder. A 30×30 grid was used and the viscous equations were integrated to $t = 2.0 \times 10^{-3}s$. No variation was seen in the flow field.

y	x_{min}	x_{max}
0.2	0.12	8.4
4.0	0.75	1.55
-3.0	0.55	1.48

Table 1: Argument ranges for 1% error.

B Approximate Power Routine

When computing $z = x^y$, we can break the calculation into stages and compute $z = exp[y\ ln(x)]$. For x reasonably close to 1, we may approximate the logarithm with

$$ln(x) \simeq 2\left[r + \frac{1}{3}r^3 + \frac{1}{5}r^5 + \frac{1}{7}r^7\right] \quad , \tag{80}$$

where

$$r = \frac{x-1}{x+1} \quad . \tag{81}$$

The exponential is then approximated by the usual expansion

$$e^t \simeq 1 + t + \frac{t^2}{2!} + \frac{t^3}{3!} + \frac{t^4}{4!} + \frac{t^5}{5!} \quad . \tag{82}$$

This procedure produces results with less than 1% error for the arguments shown in Table 1. The floating-point operation count is 30. Note that the exponent in equation (24) is fairly small over the range of effective specific heats expected for most gasdynamic situations and the approximation is used if the base ($x = P_L/P_R$) is within the first range shown in Table 1. However, the exponent in equation (21) is usually large ($5 < y < 12$) and so, the approximation is applied to equation (21) as $z = x^8\ x^{y-8}$.

C Header File

```
/* cns4u.h
 * Header file for the flow solver code cns4u.c.
 *
 * NOTE...
 * ----
 * We need to include the files "compilers.h" and "physics.h"
 * before this file.
 */

#define  DEBUG  1
/*
 * Debugging level...
 * 0 = no debugging
 * 1 = print message on entry to infrequently used functions
 * 2 = print message on entry to frequently used functions
 * 3 = dump data every step
 *
 */

/*--------------------------------------------------------------------*/


              /*******************************/
              /* Data Structure Definitions */
              /*******************************/

/*
 * --------------------------
 * types of boundary conditions for blocks
 * ADJACENT   adjacent to another tile
 * SUP_IN     supersonic inflow
 * SUP_OUT    supersonic outflow
 * SLIP       slip/tangency (adiabatic)
 * ADIABATIC  no-slip, adiabatic
 * FIXED_T    no-slip, fixed T wall
 *
 * SPECIAL    special purpose code has been included in
 *            the routines apply_inviscid_bc() and
 *            apply_viscous_bc().
 */

#define    ADJACENT  0
#define    SUP_IN    1
#define    SUP_OUT   2
#define    SLIP      3
#define    ADIABATIC 4
#define    FIXED_T   5

#define    SPECIAL  -1
```

```
/*
 * -------------------------
 * types of blocks
 * shock-layer, boundary-layer, outer-region
 */

#define INACTIVE 0
#define BL  1
#define SL  2
#define OR  3

/*
 * -------------------------
 * NL        : number of levels in the time-stepping procedure
 * DATA_DIM  : spatial dimensions of data
 */
#define    NL        2
#define    DATA_DIM  2



typedef struct cell_center
   {
            /* GEOMETRY */
   double X, Y;           /* Centre coordinates, m          */
   double volume;         /* Cell volume (unit depth), m**3 */
   double area;           /* (x,y)-plane area, m**2         */

            /* PRIMARY VARIABLES */
   double rho;            /* density, kg/m**3               */
   double u;              /* normal velocity, m/s           */
   double v, w;           /* tangential velocities          */
                          /* nominal directions for u,v,w   */
                          /* are x,y,z respectively         */
   double e;              /* specific internal energy, J/kg */
   double p;              /* pressure, Pa                   */
   double a;              /* speed of sound, m/s            */
   double T;              /* temperature, K                 */
   double mu;             /* viscosity, Pa.s                */
   double f[NSPECD];      /* species mass fractions         */

           /* CONSERVED VARIABLES */
           /* mass and specied appear above */
   double ru;             /* X-momentum/unit volume         */
   double rv;             /* Y-momentum/unit volume         */
   double rE;             /* Total Energy/unit volume       */

   double DrDt[NL];       /* updates for density/mass       */
   double DruDt[NL];      /* X-momentum                     */
   double DrvDt[NL];      /* Y-momentum                     */
   double DrEDt[NL];      /* Total Energy                   */
   double DfDt[NL][NSPECD]; /* Species mass fractions       */

            /* PRODUCTION VECTOR */
   double Q_r;            /* Mass production from sources   */
   double Q_ru;           /* X-Momentum from body forces    */
   double Q_rv;           /* Y-Momentum from body forces    */
   double Q_rE;           /* Total Energy production        */
   double Q_rf[NSPECD];   /* Species production; reactions  */
   };
```

```
typedef struct cell_interface
   {
            /* GEOMETRY */
   double length;          /* Interface length in the x,y-plane */
   double cosX, cosY;      /* Directio. cosines for unit normal */

            /* PRIMARY VARIABLES */
   double u;               /* normal velocity, m/s              */
   double v;               /* tangential velocities             */
   double T;               /* temperature, K                    */
   double k;               /* heat flux coefficient             */
   double mu, lambda;      /* coefficients of viscosity         */

            /* FLUXES OF CONSERVED QUANTITIES */
   double F_r,  G_r;       /* Mass / unit time / unit area   */
   double F_ru, G_ru;      /* X-momentum                     */
   double F_rv, G_rv;      /* Y-momentum                     */
   double F_rE, G_rE;      /* Total Energy                   */
   double F_rf[NSPECD];    /* Species mass fractions (X-comp)*/
   double G_rf[NSPECD];    /* Species mass fractions         */
   };



typedef struct cell_vertex
   {
            /* GEOMETRY */
   double X, Y;            /* Coordinates, m                      */
   double area;            /* x,y-plane area of secondary cells   */

            /* DERIVATIVES OF PRIMARY VARIABLES */
   double dudx, dudy, dvdx, dvdy;   /* velocity derivatives     */
   double dTdx, dTdy;               /* Temperature deriv.       */
   };



/*-----------------------------------------------------------------*/

               /***********************************/
               /* THE SINGLE-BLOCK Data Structure */
               /***********************************/

typedef struct block_data
   {
   /*
    * This data structure should contain everything needed for
    * a single-block solution -- both geometry and flow data
    * There are a few references to MULTI-BLOCK data but they
    * should not affect things unless the macro "MULTI_BLOCK"
    * is defined.
    */

#  ifdef MULTI_BLOCK
   struct node_of_graph *mynode; /* node that owns this data   */
#  endif
```

```
double dt;                      /* magnitude of time step, s  */
double dt0;                     /* initial time step          */
double dt_allow;                /* Allowable time step        */
double cfl_target;              /* desired CFL number         */
double sim_time;                /* simulation time in seconds */
double cfl_min, cfl_max;        /* estimates of CFL number    */
double cfl_tiny, time_tiny;     /* the smallest so far...     */
int max_steps;                  /* max number of time steps   */
                                /* on this grid               */

int Xorder;                     /* spatial order 1 or 2       */
int Torder;                     /* temporal order 1 or 2      */

double tplot, dtplot;           /* timer for writing solution */
double this, dthis;             /* another sample timer       */
int hncell;                     /* number of sample cells     */
int hxcell[NDIM], hycell[NDIM]; /* location of sample cell    */

struct flow_state free_str;     /* free-stream properties     */
struct flow_state init_str;     /* initial flow properties    */
                                /* These are used to set up   */
                                /* uniform flow conditions.   */

int nxdim, nydim;
/*
 * Total number of cells in exch direction for this block.
 * these will be used in the array allocation routines.
 */

int nnx, nny;
/* Number of active cells in the X,Y-directions */

int nghost;
/* Number of ghost cells around the boundaries. */

int ixmin, ixmax;
int iymin, iymax;
/*
 * These index limits are set to allow convenient access
 * to the arrays without having to worry how many buffer
 * cells are present.
 * ixmin <= ix <= ixmax,  iymin <= iy <= iymax
 * Typically ixmin = iymin = 2.
 */

int nsp;
/* Number of species (1 <= nsp <= NSPECD) */

int bc_N, bc_S, bc_E, bc_W;
/*
 * Boundary condition flags for the North, South, East and West
 * block-domain boundaries.  Options are ADJACENT, SUP_IN, SUP_OUT,
 * SLIP, ADIABATIC, and FIXED_T
 */

double Twall_N, Twall_S, Twall_E, Twall_W;
/*
 * Wall temperatures for use with the FIXED_T boundary condition.
 * (NOTE: Only relevant for viscous flows)
 */
```

```
   int region_type;
   /*
    * type of region this is (INACTIVE, SL, BL, or OR)
    */

   int viscous;
   /*
    * viscous=0:  inviscid terms only
    *         1:  include viscous terms
    */

   int axisymm;
   /*
    * axisymm=0:  2D planar equations
    *         1:  2D axisymmetric equations
    */

   double delta[DATA_DIM+1];
   /*
    * discretization parameters (delta x, delta y, and delta t)
    */

   struct cell_center    **Ctr;
   struct cell_interface **VF;
   struct cell_interface **HF;
   struct cell_vertex    **Vtx;
   /*
    * Most of the data is stored in the preceding arrays.
    * Ctr[ix][iy] = cell center values
    * VF[ix][iy]  = vertical face properties
    * HF[ix][iy]  = horizontal face properties
    * Vtx[ix][iy] = cell vertex values
    *
    * VF and HF are used to interface to the Riemann solver
    * and to store the interface fluxes.
    *
    * Vtx is used for the viscous terms.
    */

   };  /* end of THE data structure definition */

/*-----------------------------------------------------------------*/
```

```
/*
 * --------------------
 * Indexing of the data.
 * --------------------
 *
 * The following figure shows cell [ix,iy] and its associated
 * vertices and faces.
 *
 *
 *
 *                        North (horizontal)
 *     Vertex C           face                    Vertex B
 *     [ix-1,iy]          [ix,iy]                 [ix,iy]
 *              +-------------x--------------+
 *              |                            |
 *              |                            |
 *              |                            |
 *              |                            |
 *              |                            |
 *              |                            |
 *   West       |          cell center       |  East (vertical)
 *   face       |            [ix,iy]         |  face
 *   [ix-1,iy] x               o             x  [ix,iy]
 *              |                            |
 *              |                            |
 *              |                            |
 *              |                            |
 *              |                            |
 *              |                            |
 *              |                            |
 *              |                            |
 *              +-------------x--------------+
 *     Vertex D            South               Vertex A
 *     [ix-1,iy-1]         face                [ix,iy-1]
 *                         [ix,iy-1]
 *
 *
 * Thus...
 * ----
 * Active cells are indexed as Ctr[ix][iy], where
 * ixmin <= ix <= ixmax, iymin <= iy <= iymax.
 *
 * Acitve vertical interfaces are indexed as VF[ix][iy], where
 * ixmin-1 <= ix <= ixmax, iymin <= iy <= iymax.
 *
 * Acitve horizontal interfaces are indexed as HF[ix][iy], where
 * ixmin <= ix <= ixmax, iymin-1 <= iy <= iymax.
 *
 * Active vertices are indexed as Vtx[ix][iy], where
 * ixmin-1 <= ix <= ixmax, iymin-1 <= iy <= iymax.
 *
 * Space for ghost cells is available outside these ranges but
 * within the dimensioned ranges
 * 0 <= ix <= nxdim-1, 0 <= iy <= nydim.
 *
 * ------------------------------------------------------------ */
```

```
/*
 * ---------------
 * Indexing Macros
 * ---------------
 *
 * These macros should make indexing over the A,B,C,D vertices
 * and over the North,South,East,West faces more readable.
 * NOTE that they are defined with respect to the [ix,iy] cell.
 */


/* VERTICES */

#define  ixA   (ix)
#define  iyA   (iy-1)

#define  ixB   (ix)
#define  iyB   (iy)

#define  ixC   (ix-1)
#define  iyC   (iy)

#define  ixD   (ix-1)
#define  iyD   (iy-1)


/* CELL FACES */

#define  ixN   (ix)
#define  iyN   (iy)

#define  ixS   (ix)
#define  iyS   (iy-1)

#define  ixE   (ix)
#define  iyE   (iy)

#define  ixW   (ix-1)
#define  iyW   (iy)
```

```
/*=================================================================*/

/*
 * ------------------
 * Multi-block MACROs
 * ------------------
 *
 * These are used in the boundary condition routines when
 * copying data from multi-block buffers to ghost-cells.
 *
 */

#ifdef MULTI_BLOCK

/*
 * Directions that neighbors can have
 *
 * when to_direction is NE, SE, NW, SW then
 * to_direction = mod(from_direction+2,4) + 4
 *
 * when to_direction is N, S, E, W then
 * to_direction = mod(from_direction+2,4)
 *
 * Here is a stencil ...   and the indexing equivalent
 *
 *          N                          (i  ,j+1)
 *          |                              |
 *          |                              |
 *      W --+-- E      (i-1,j  ) -- (i  ,j  ) -- (i+1,j  )
 *          |                              |
 *          |                              |
 *          S                          (i  ,j-1)
 *
 */

#define NORTH     0
#define EAST      1
#define SOUTH     2
#define WEST      3
#define NORTHWEST 4
#define SOUTHWEST 5
#define NORTHEAST 7
#define SOUTHEAST 6


#endif
```

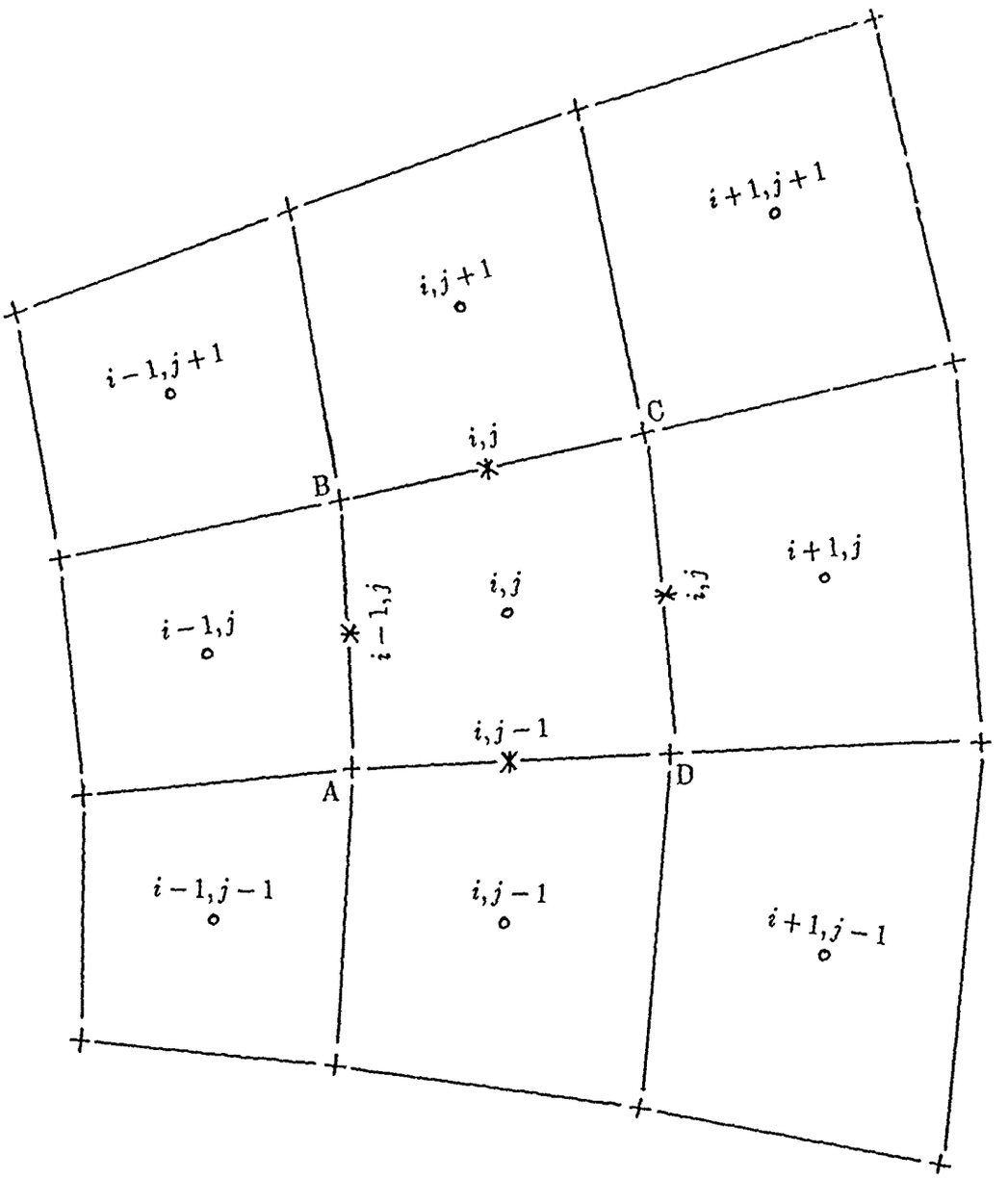


Figure 1: Finite-volume cell and indexing convention: "o" denotes a cell centre; "+" denotes a vertex; "*" denotes an interface midpoint.

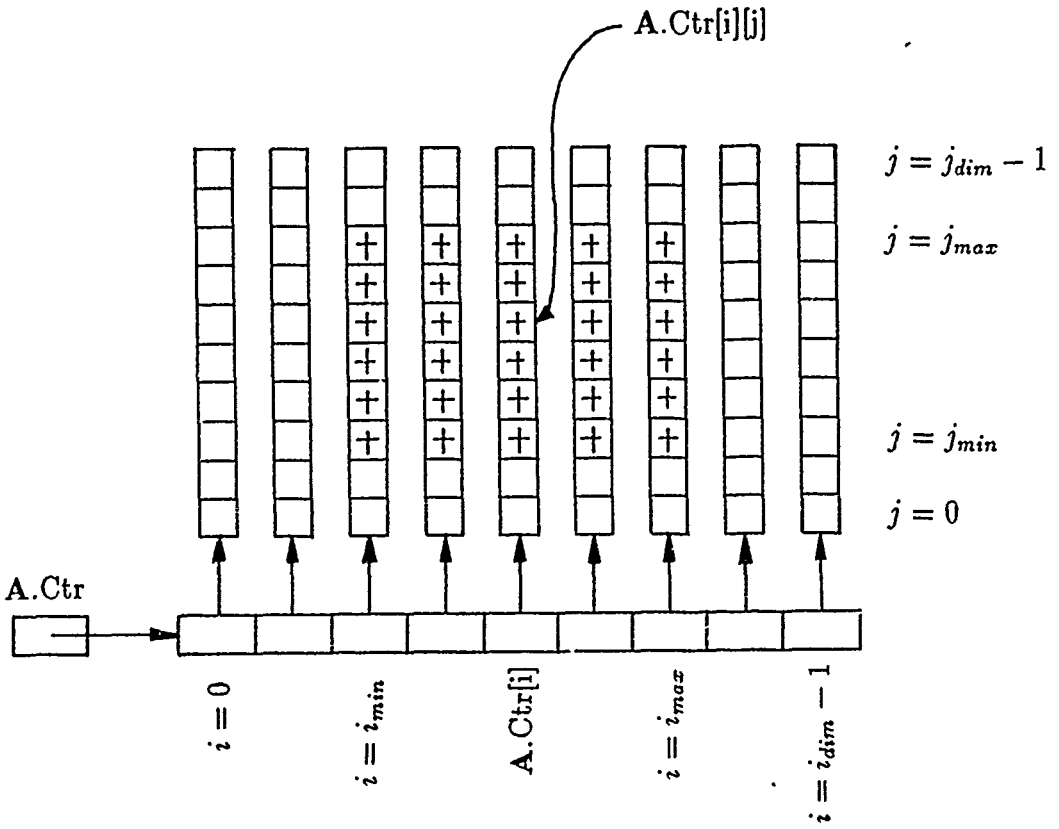


Figure 2· Storage of the data for the cell centres. Active cells are denoted by "+". Ghost cells are unmarked.

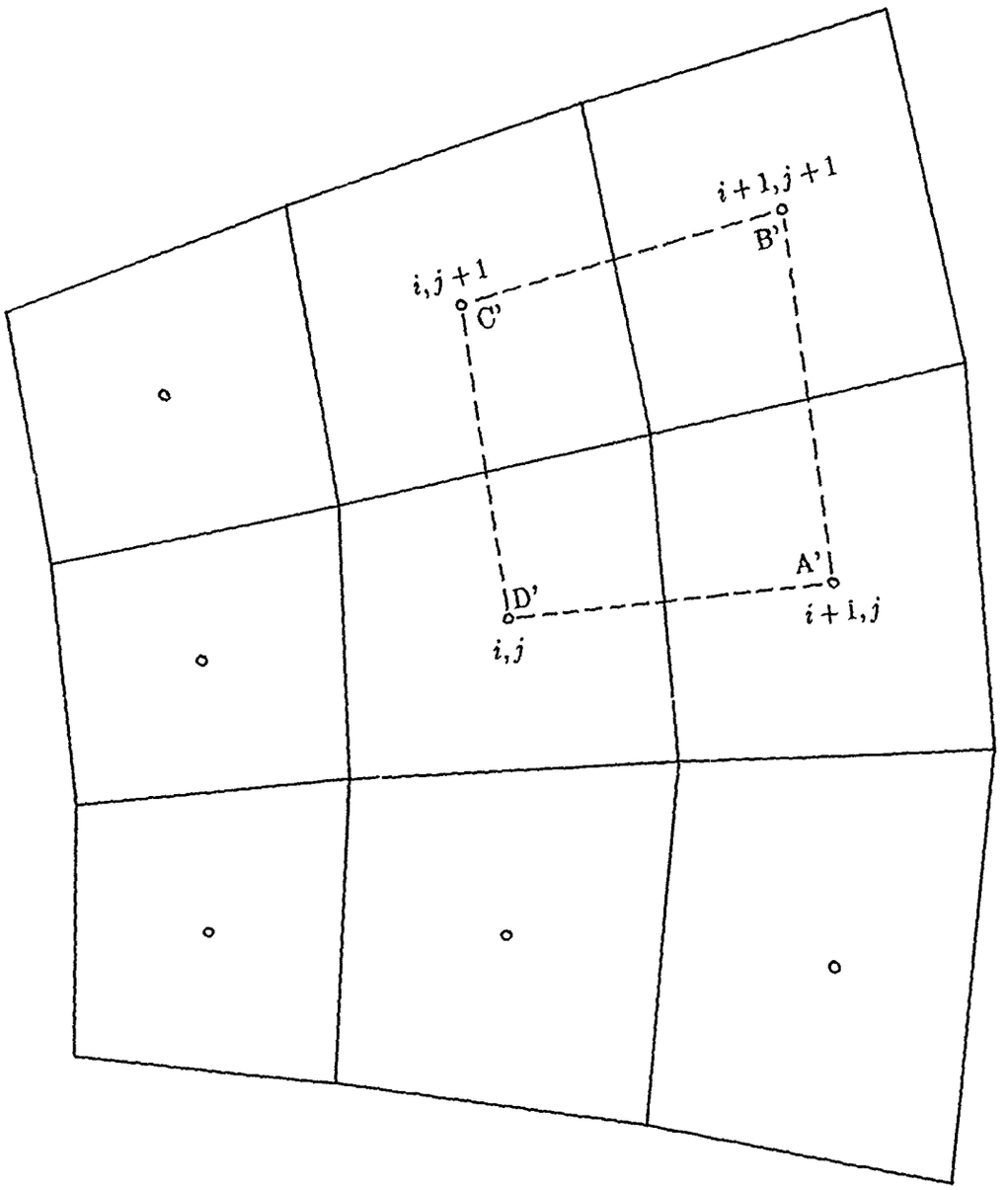


Figure 3: Secondary-cell geometry.

```
if (u* > 0) then

  The contact-discontinuity has moved to the right
  and the interface state is determined from the
  L and L* states.

  if (P* > P_L) then
    The left-moving wave is a shock.
    if (ws_L ≥ 0) then
      All waves have moved to the right.
      Interface state is equal to L.
    else
      Interface state is equal to L*.
    endif
  else
    The left-moving wave is a rarefaction.
    if (u_L − a_L ≥ 0) then
      All waves have moved to the right.
      Interface state is equal to L.
    elseif (u*_L − a*_L > 0) then
      The rarefaction straddles the interface.
      Interpolate the interface state from
      states L and L*.
    else
      The entire rarefaction moved to the
      left of the interface.
      Interface state is equal to L*.
    endif
  endif

else

  The contact discontinuity has moved to the left
  and the interface state is determined from the
  R and R* states in a similar manner...

endif
```

Figure 4: Interpolation logic for the Riemann solver.

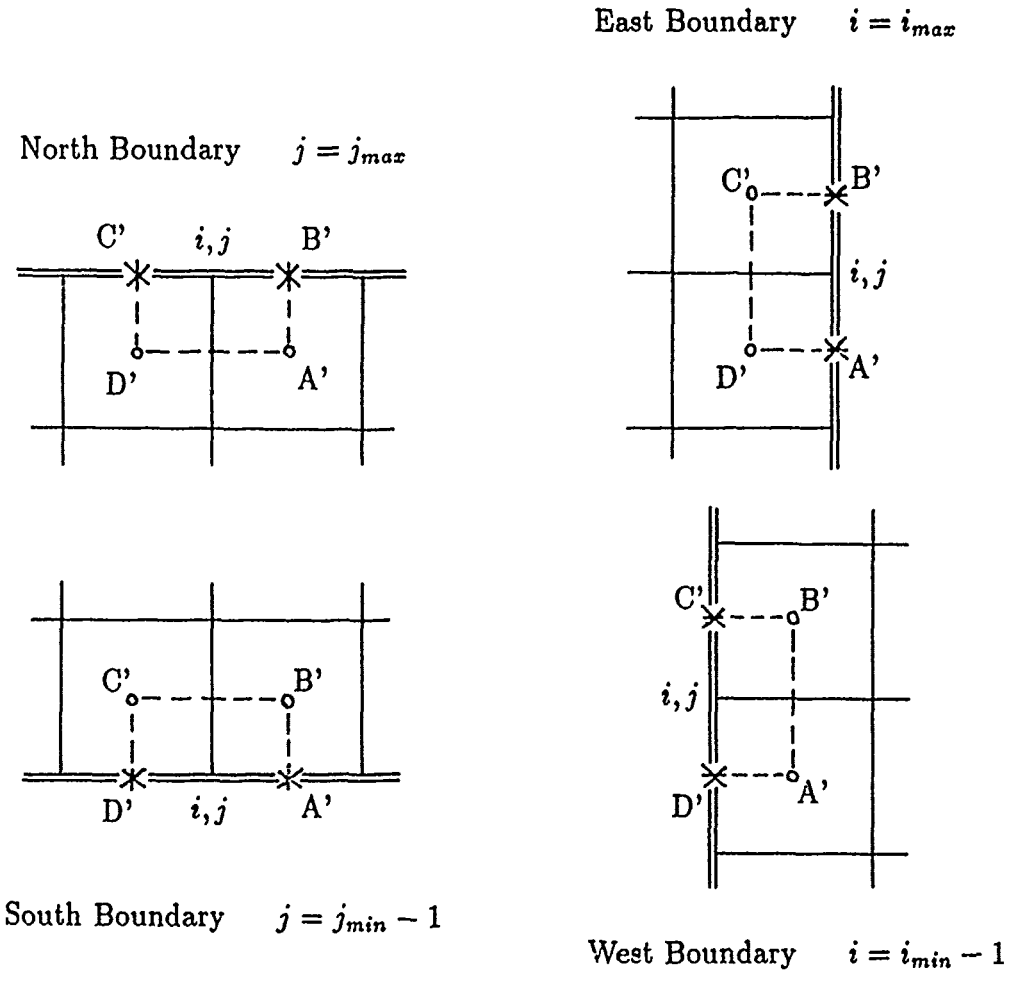


Figure 5: Definition of the secondary (half-)cells along the boundaries. The i,j vertex is identified in each case.

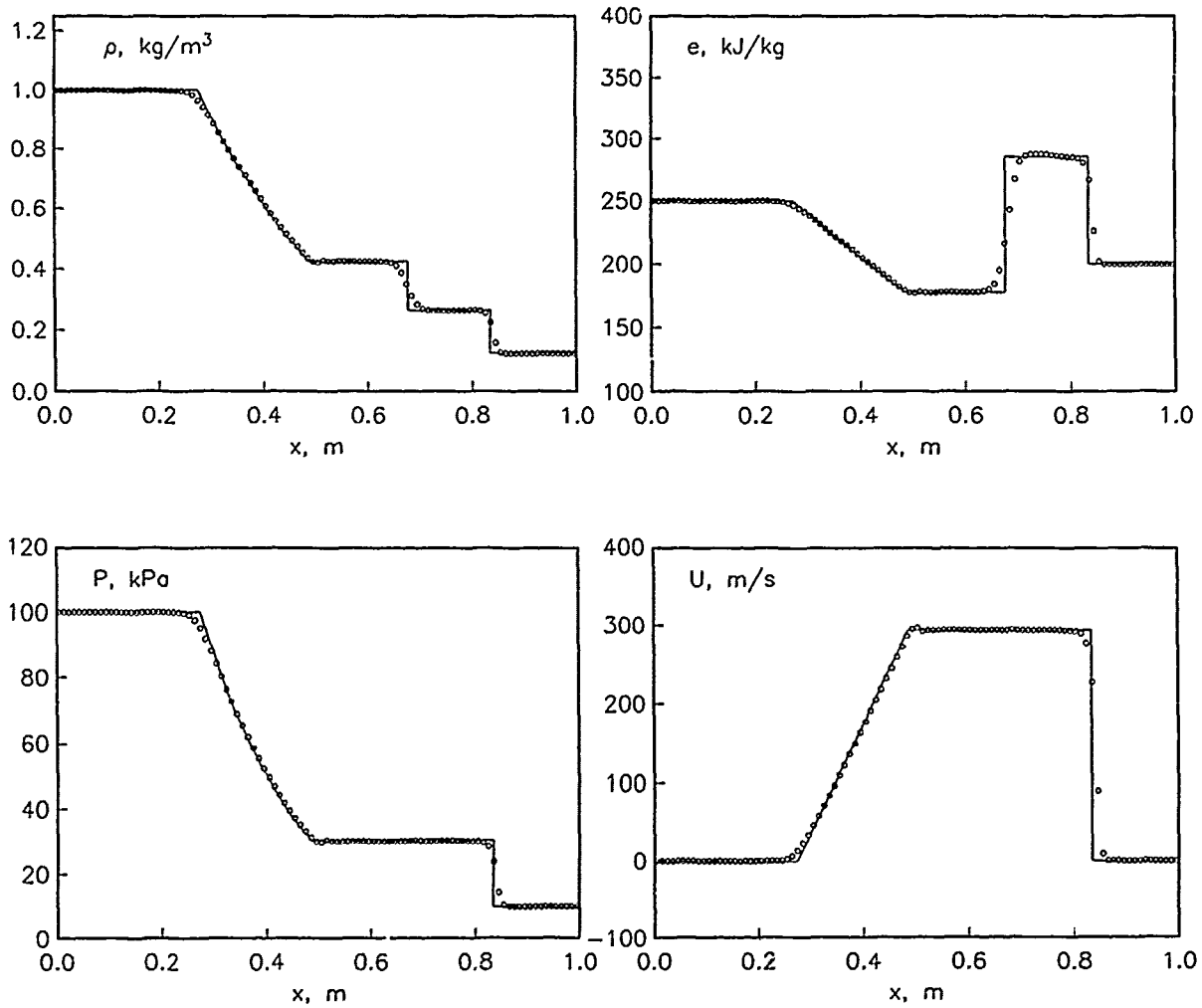


Figure 6: Flow state in the one-dimensional shock tube at $t = 6.03 \times 10^{-3}s$. Symbols denote the finite-volume solution, solid lines denote the exact solution.

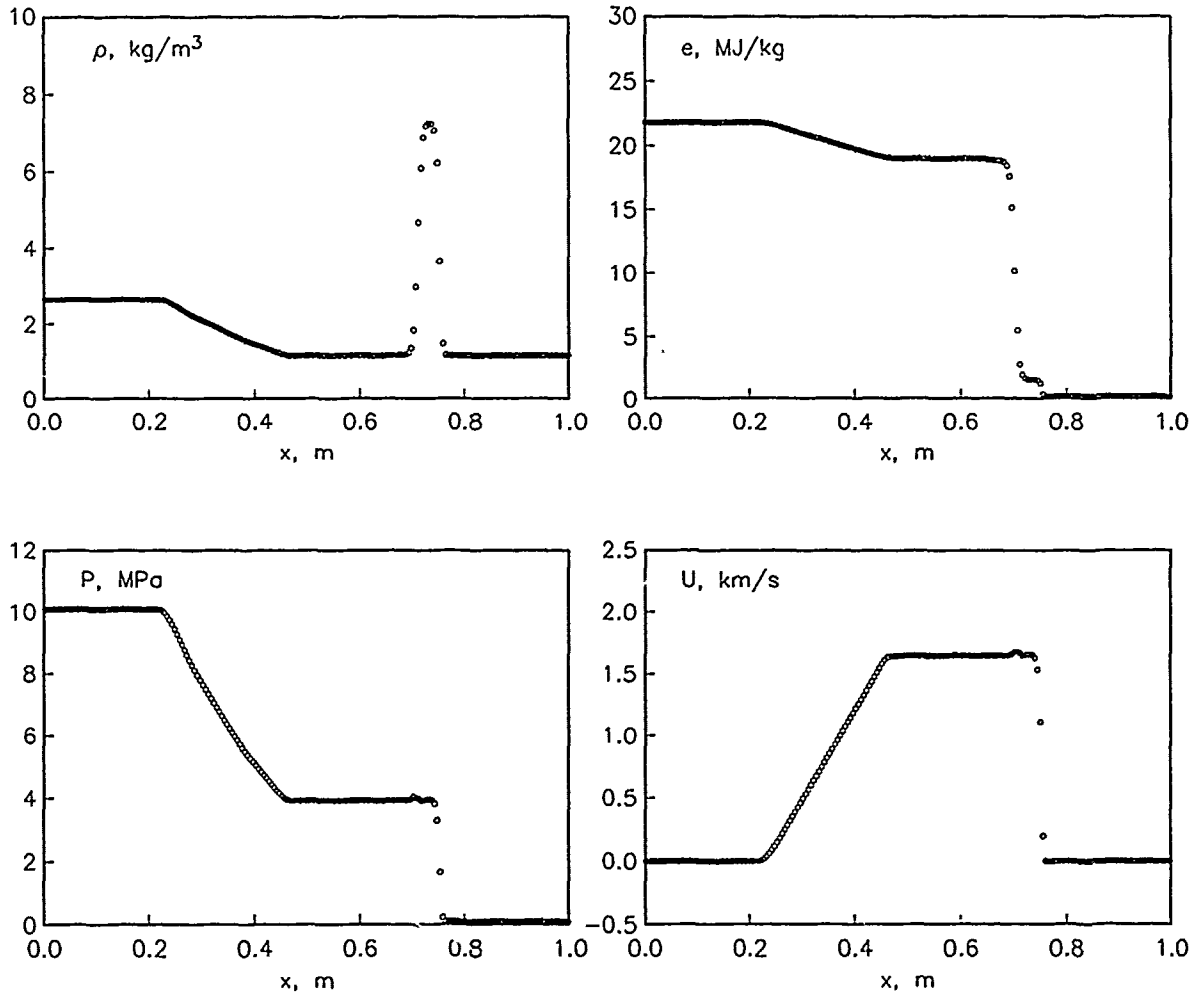


Figure 7: Flow state in the high temperature shock tube at $t = 1.25 \times 10^{-3} s$.

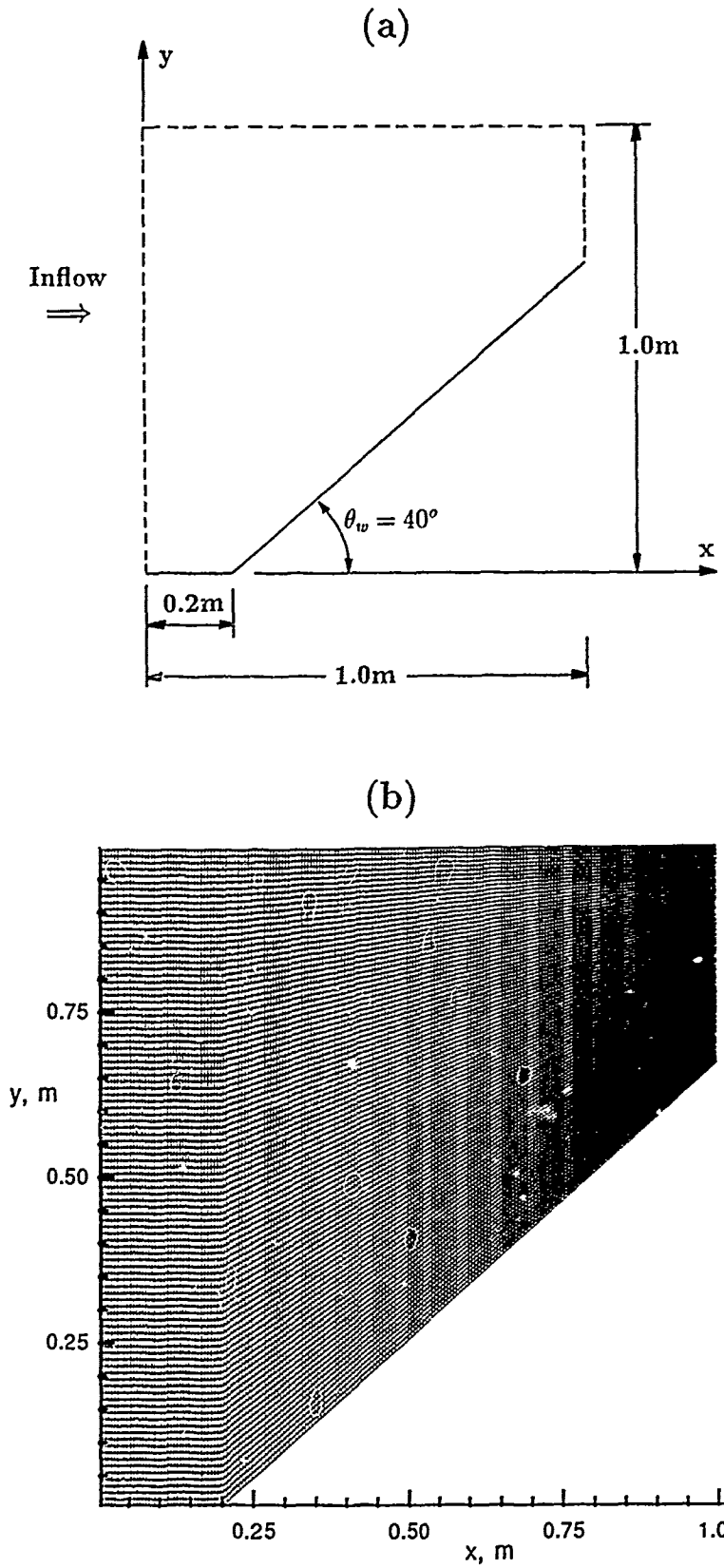


Figure 8: Flow over a wedge with a double Mach reflection: (a) flow domain; (b) 200 × 100 mesh connecting the cell centres.

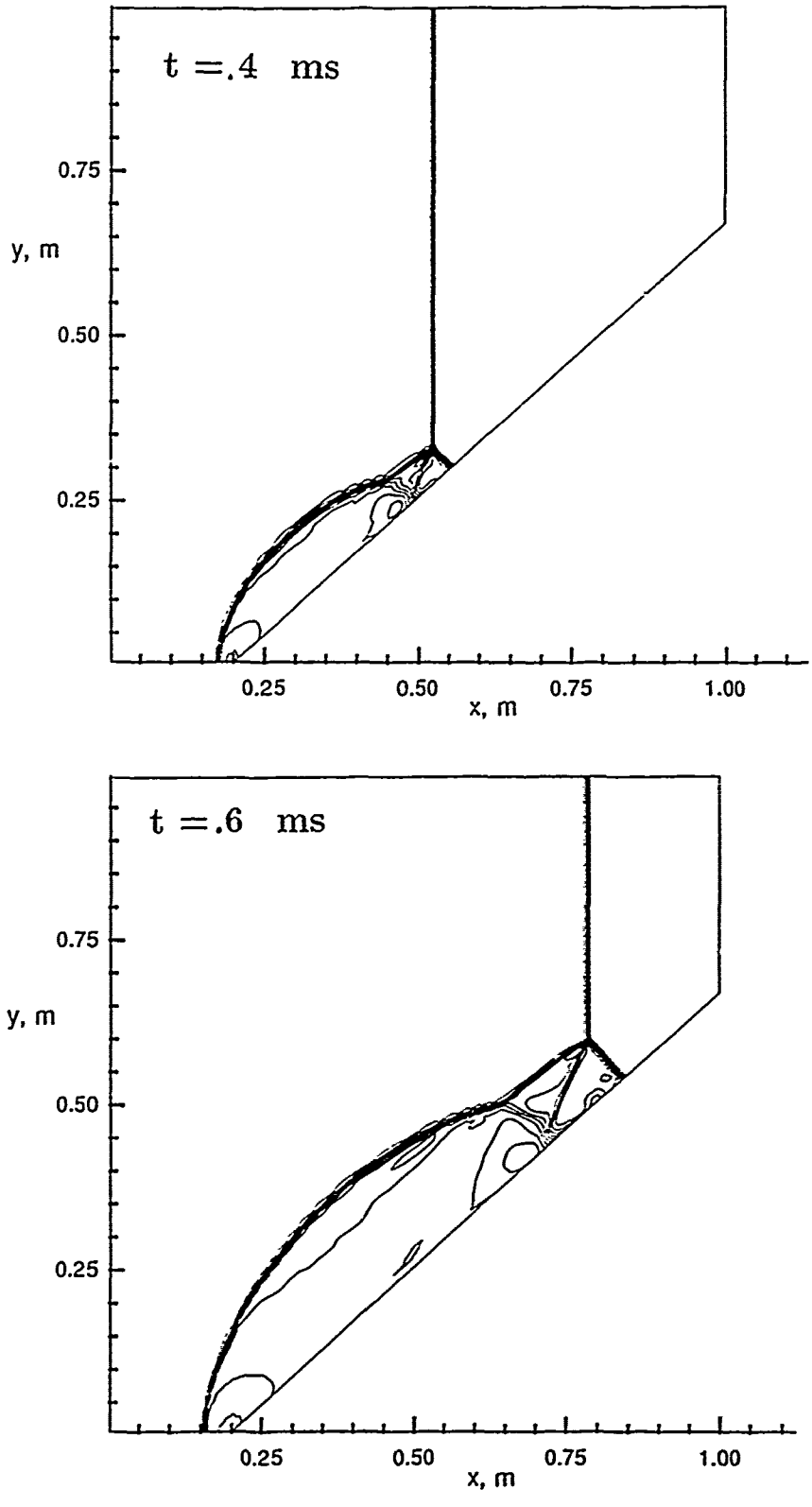
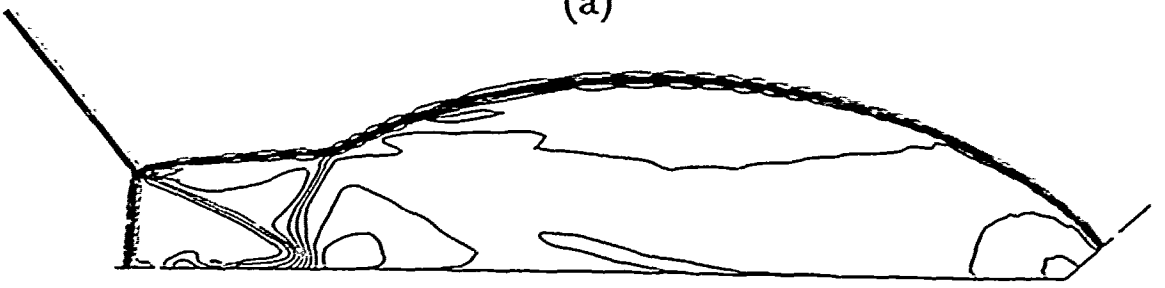
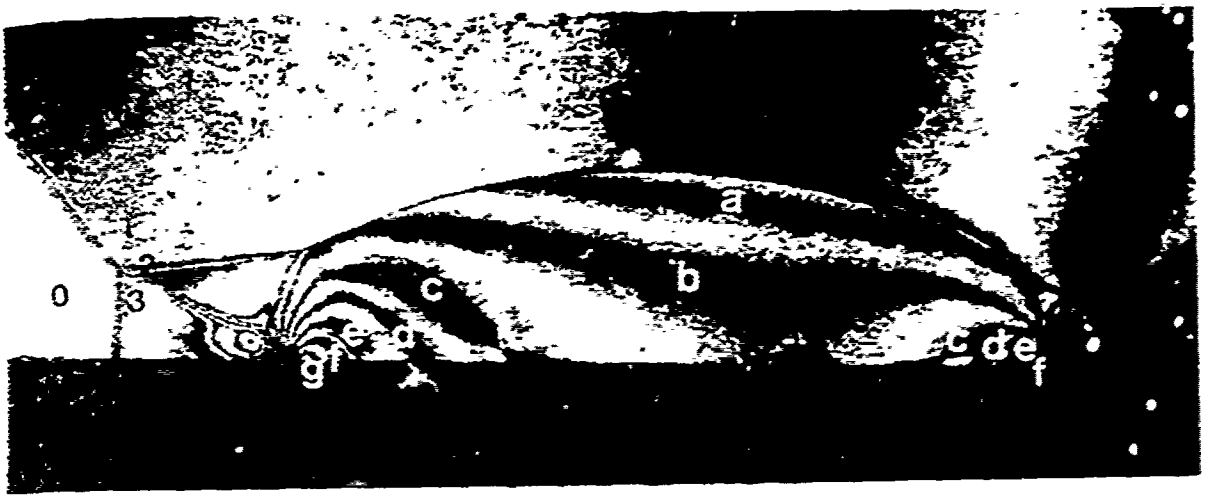


Figure 9: Density contours at two time instants for the double Mach reflection case.

(a)



(b)



Region	ρ/ρ_1
0	1.00
1	4.31
2	7.16
3	5.08
a	7.60
b	8.03
c	8.47
d	8.91
e	9.35
f	9.78
g	10.22

Figure 10: Comparison of (a) the density contours for the present solution at $t = .7$ ms and (b) an interferogram from figure (13a) in Glaz et al (1986).

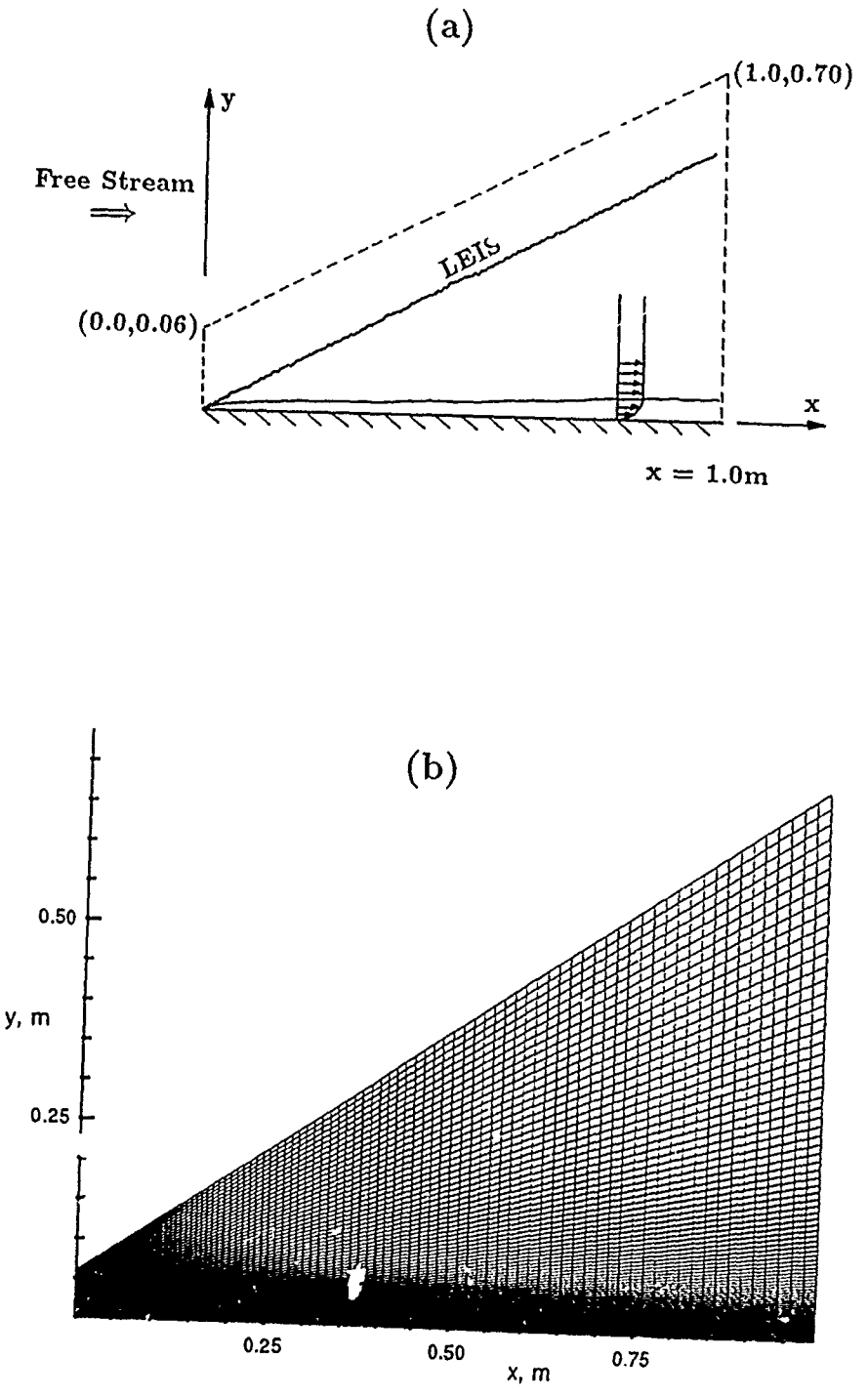


Figure 11: Boundary layer along a flat plate with $M = 2.0$ and $Re_L = 1.65 \times 10^6$: (a) flow domain; (b) 100×100 mesh joining the cell centres.

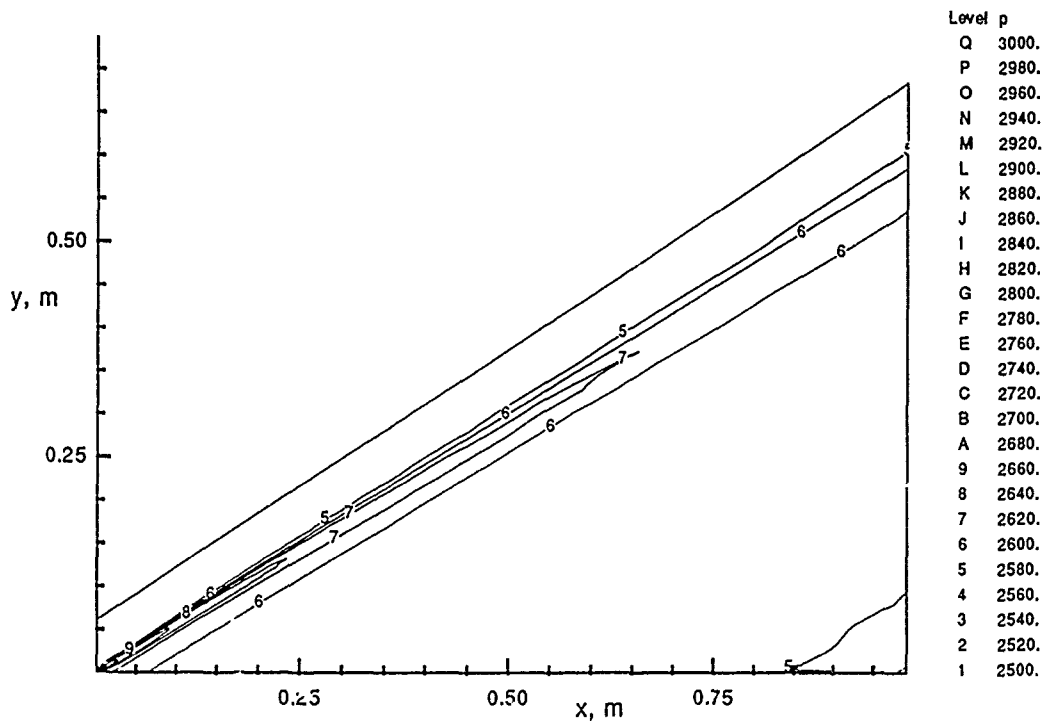


Figure 12: Pressure contours at $t = 7.0 \times 10^{-3} s$.

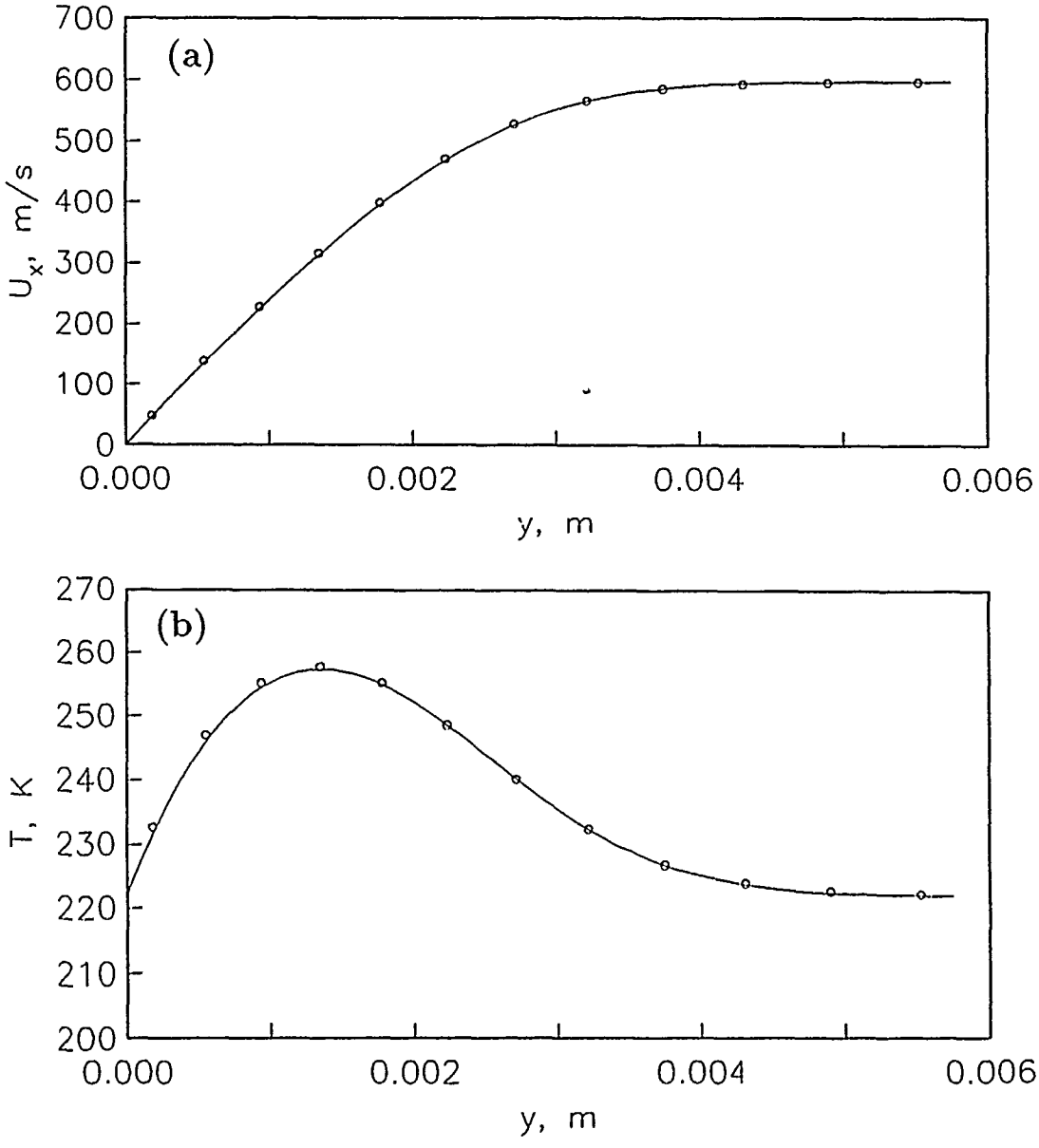


Figure 13: Comparison of the present solution (symbols) with a spectral solution (solid line): (a) x-velocity profile at $x = 0.9415m$ (i=97); (b) temperature profile at the same station.

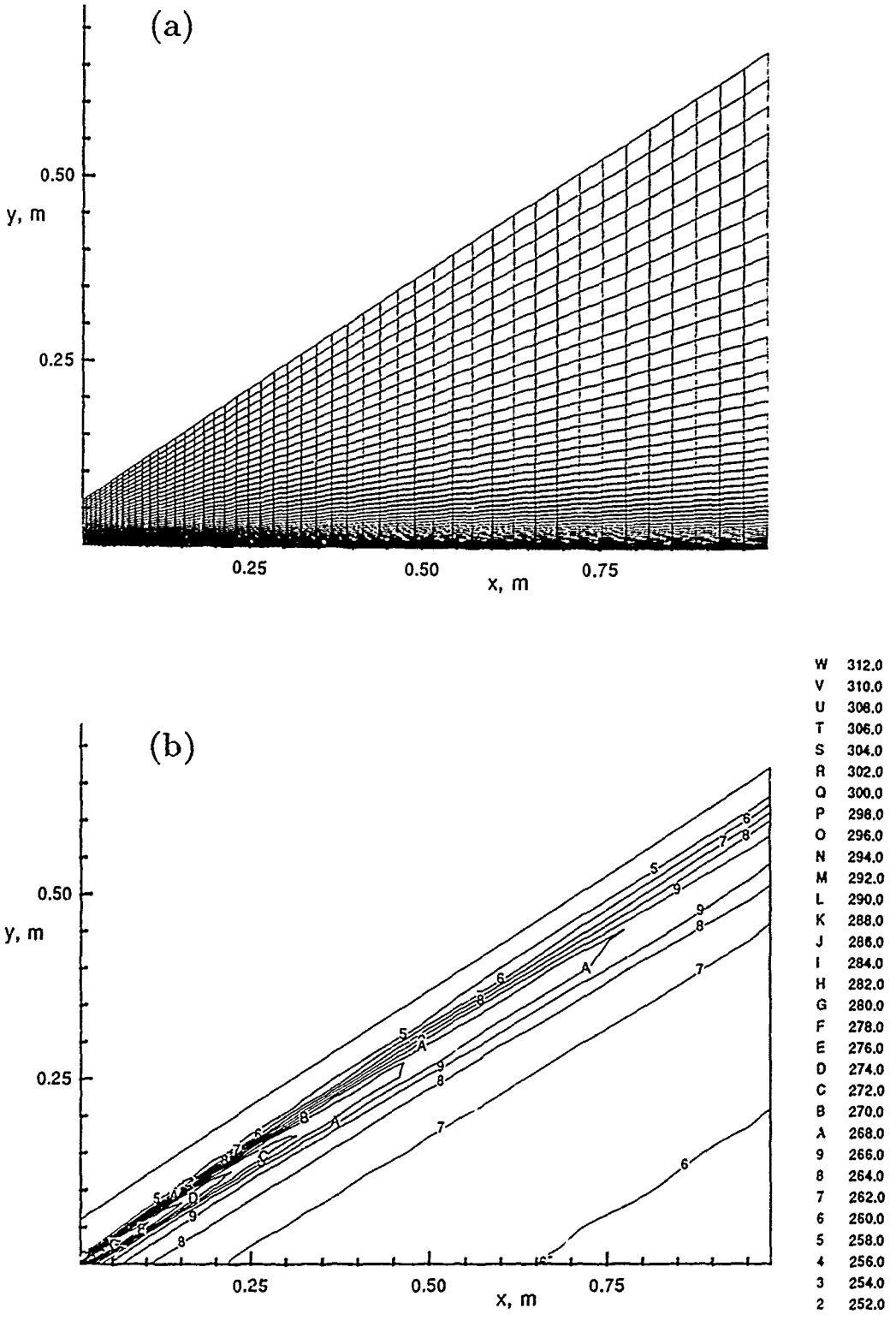


Figure 14: Boundary layer along a flat plate with $M = 2.0$ and $Re_L = 1.65 \times 10^5$: (a) 50×50 mesh joining cell centres; (b) Pressure contours at $t = 8.0 \times 10^{-3} s$.

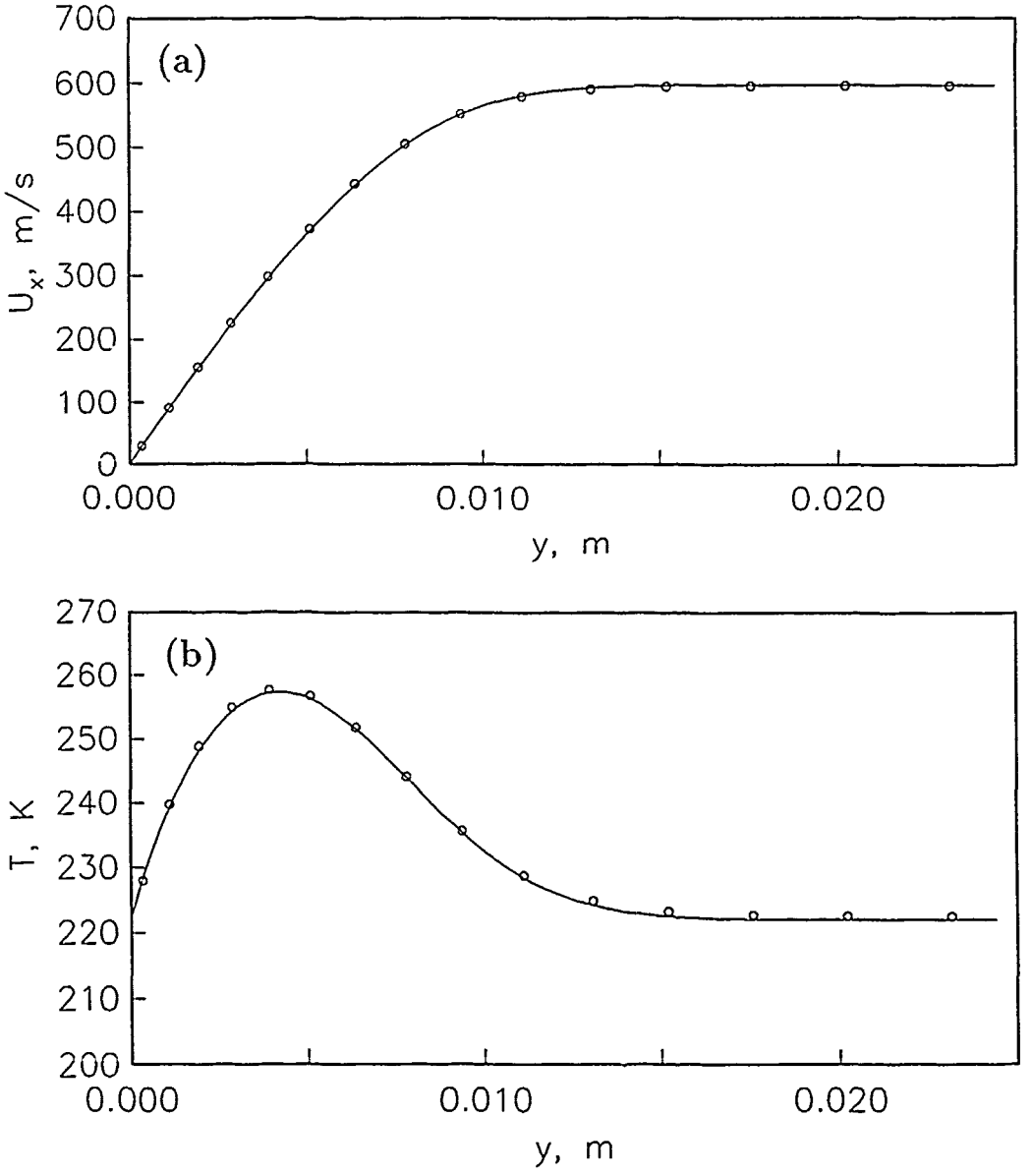


Figure 15: Comparison of the present solution (symbols) with a spectral solution (solid line): (a) x-velocity profile at $x = 0.916m$ (i=48); (b) temperature profile at the same station.

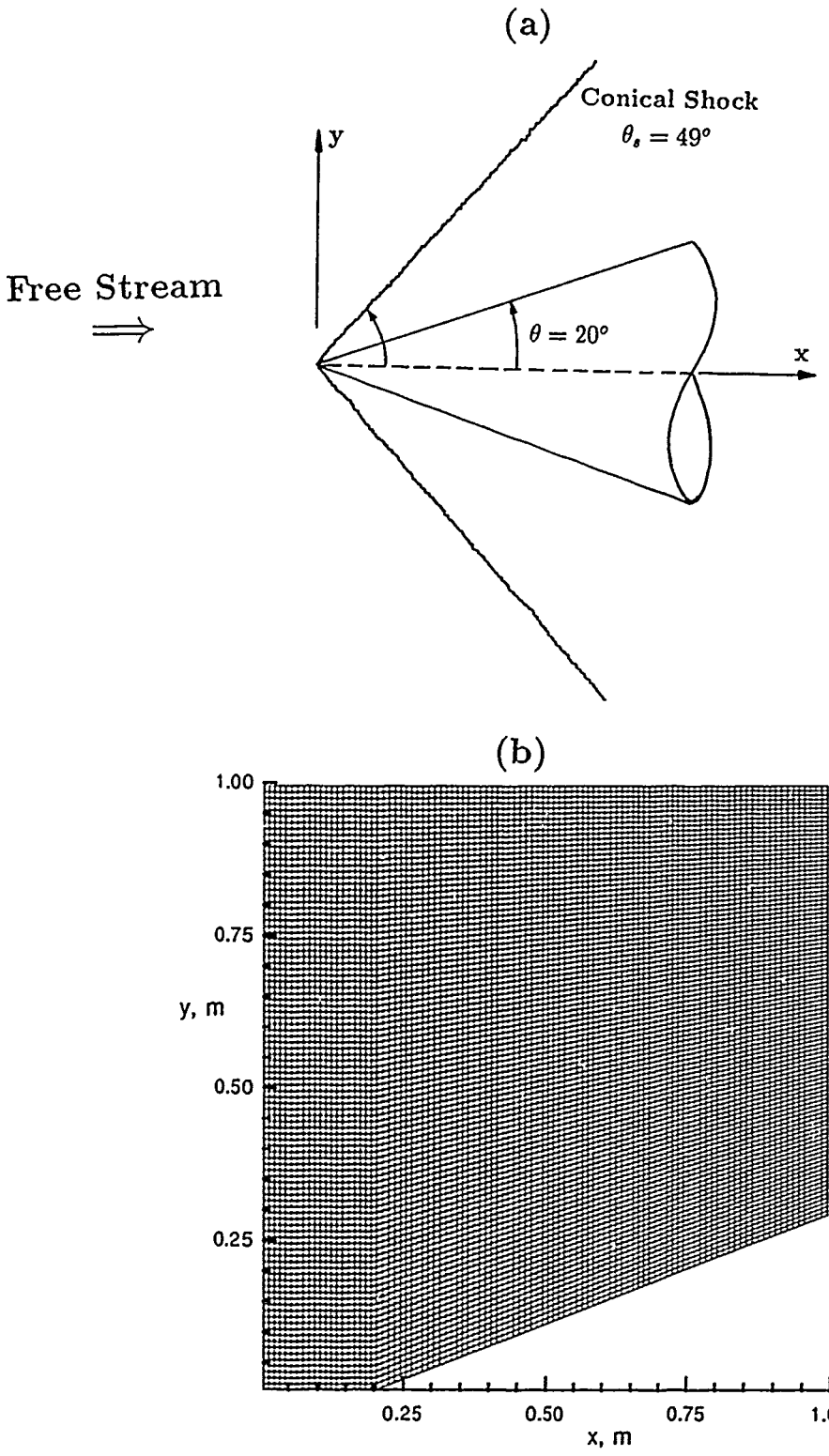


Figure 16: $M = 1.5$ inviscid flow over a 20° cone: (a) flow geometry; (b) 100 × 100 mesh joining cell centres.

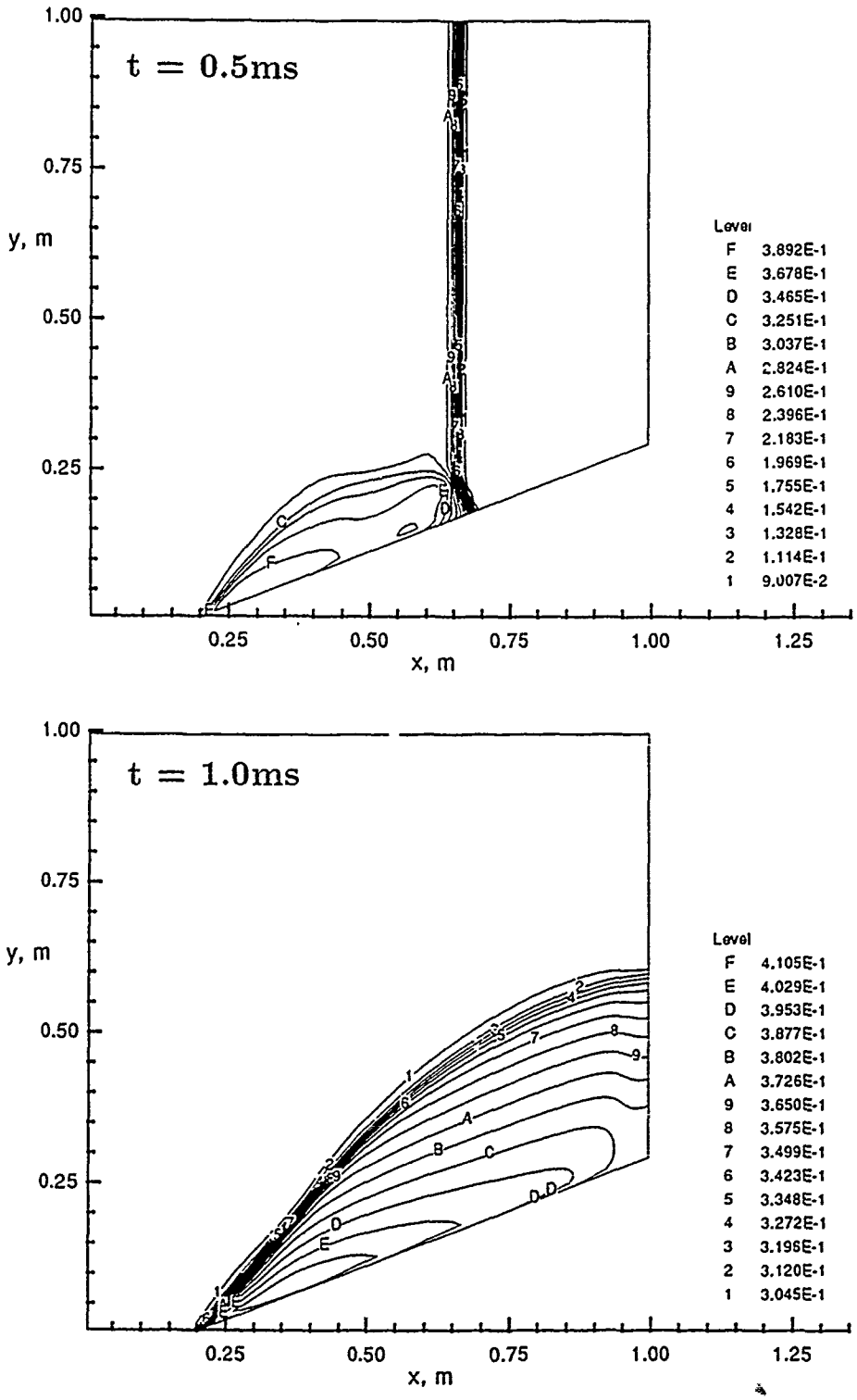


Figure 17: Early evolution of the density field over the cone.

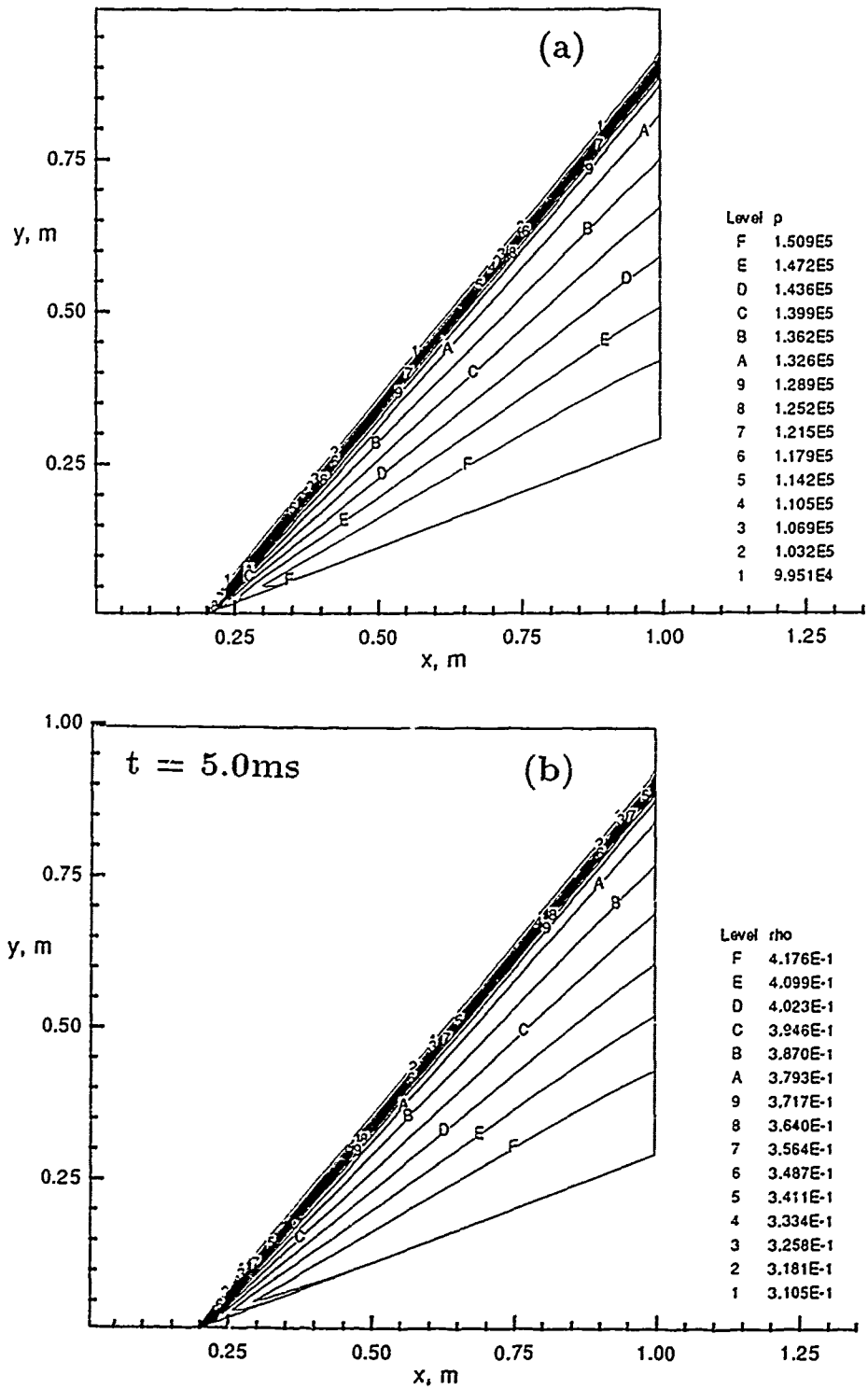
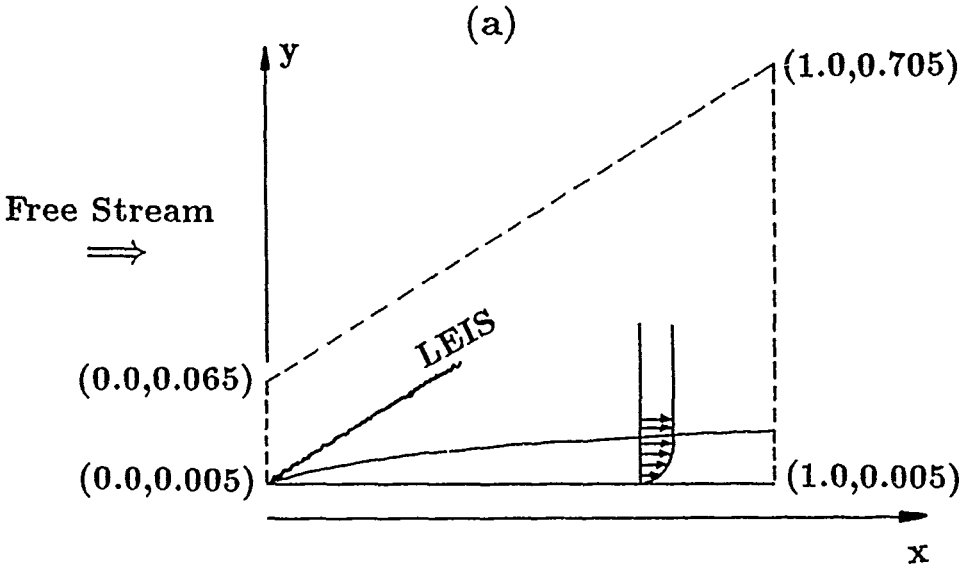


Figure 18: Flow over the cone at $t = 5.0 \times 10^{-3} s$: (a) pressure contours; (b) density contours.



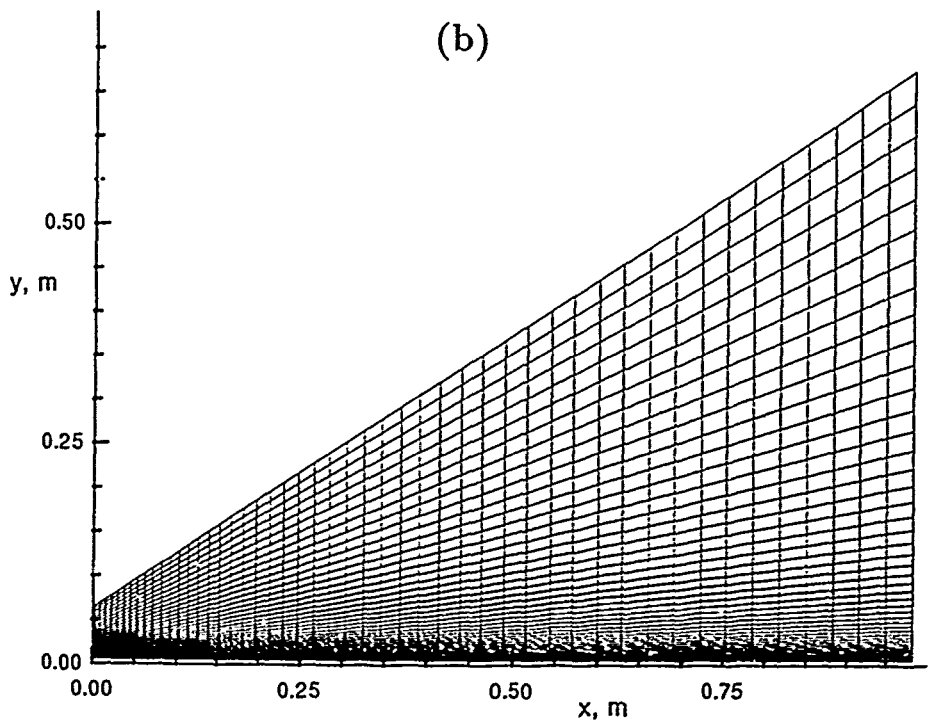


Figure 19: Boundary layer along a cylinder with $M = 2.0$ and $Re_L = 1.65 \times 10^5$: (a) flow geometry; (b) 50×50 mesh joining cell centres.

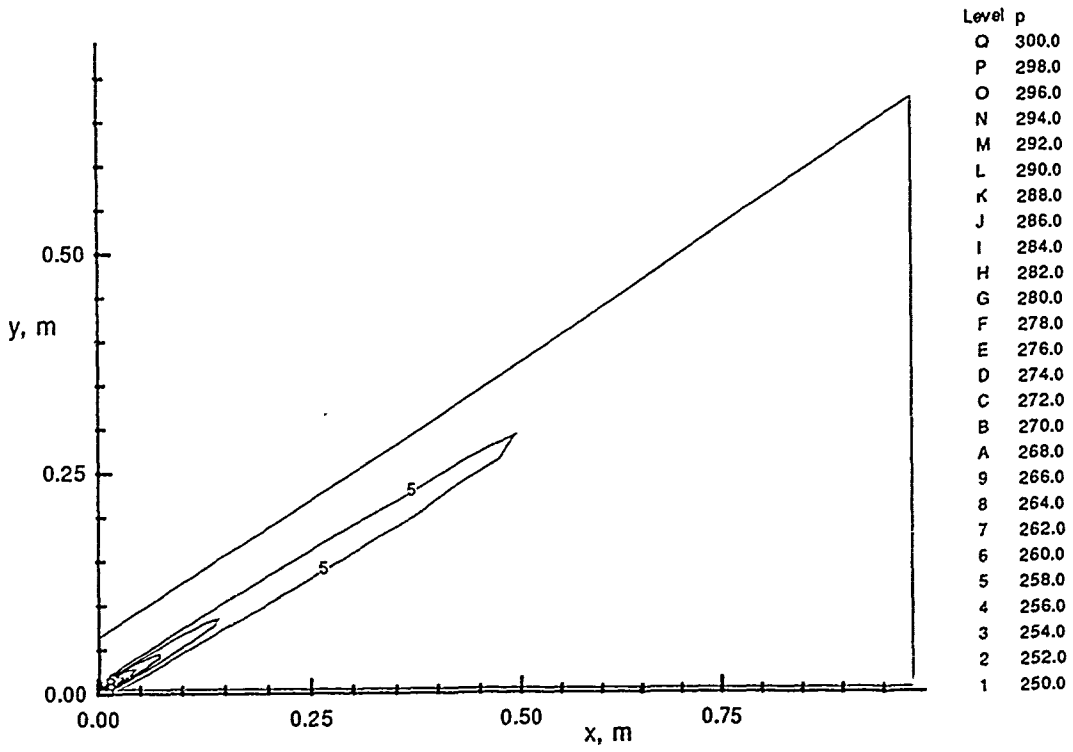


Figure 20: Pressure contours at $t = 8.0 \times 10^{-3}s$.

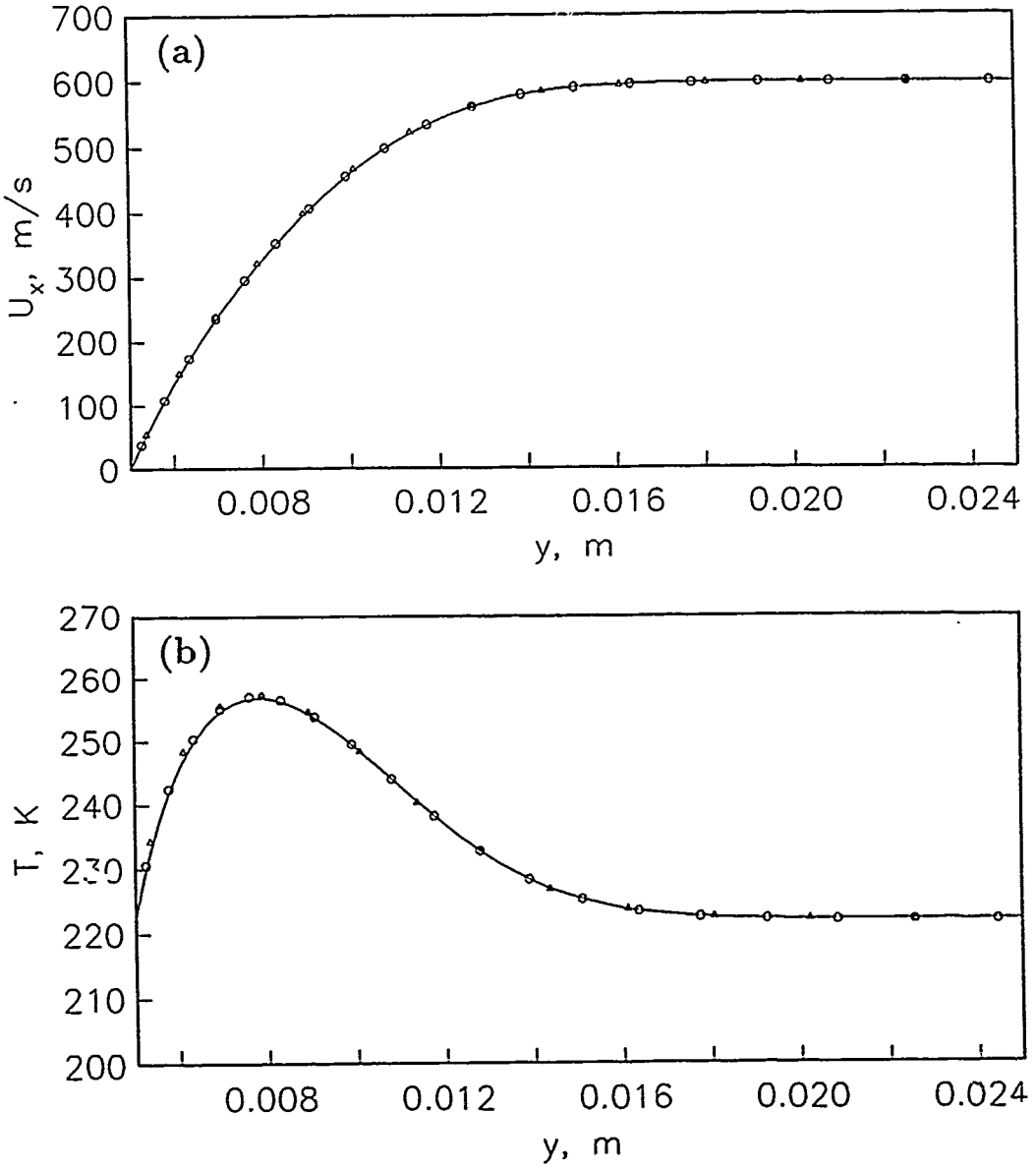


Figure 21: Comparison of the present solution ("o"= 70×70 grid, "Δ"= 50×50 grid) with a spectral solution (solid line): (a) x-velocity profile at $x = 0.916m$; (b) temperature profile at the same station.

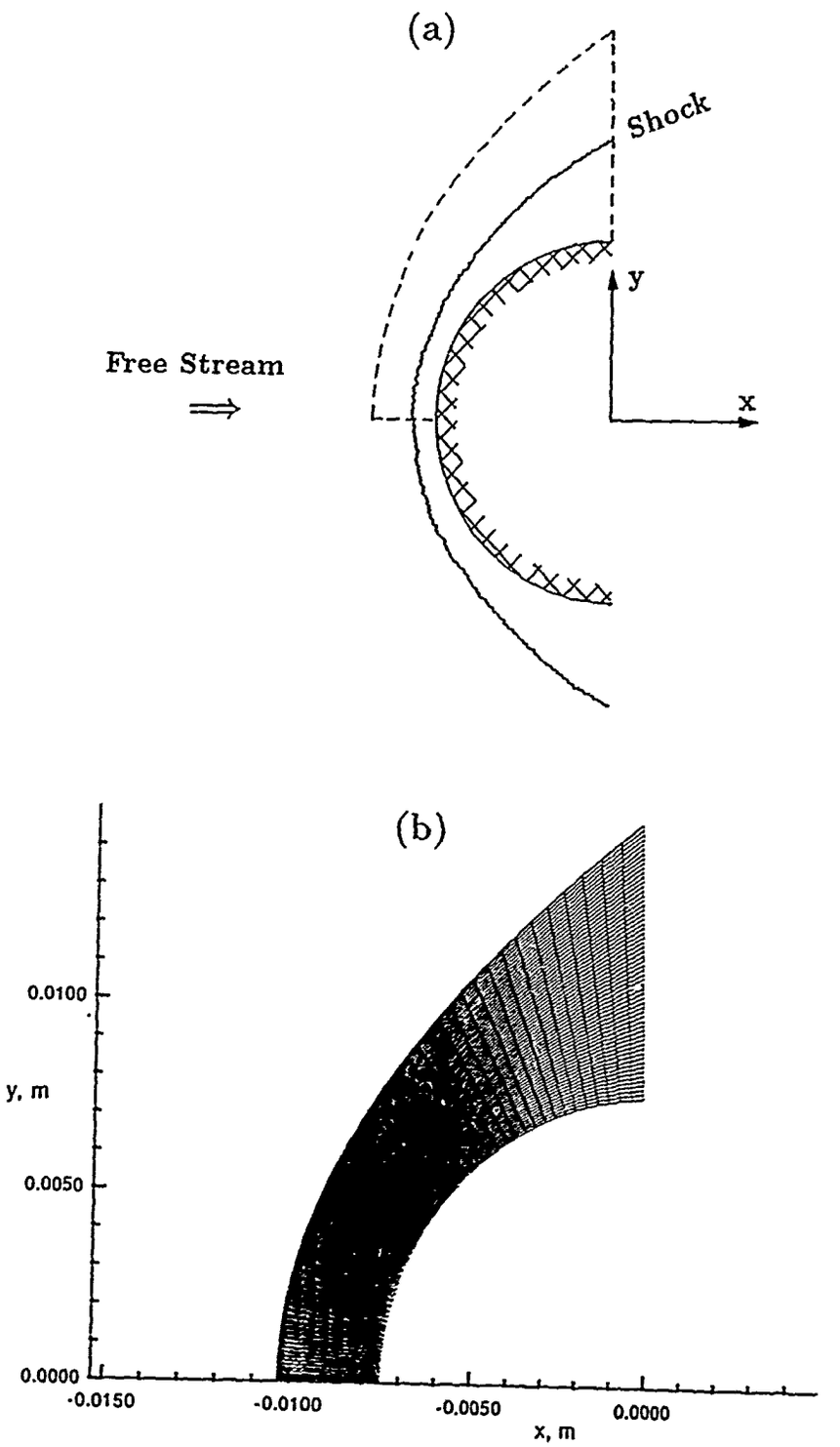


Figure 22: Sphere in a Mach 12 flow: (a) flow geometry; (b) mesh joining cell centres.

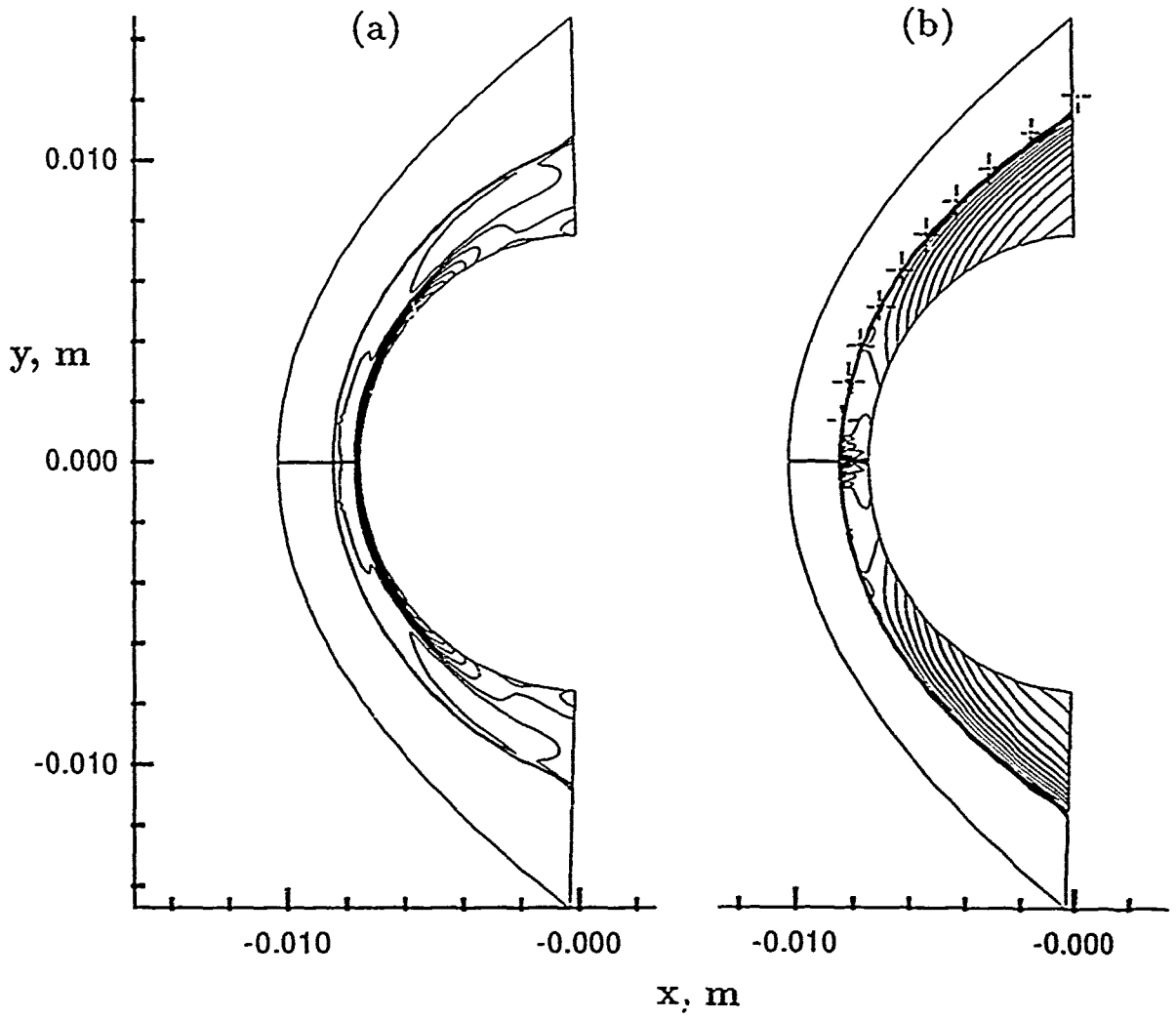


Figure 23: Evolution of the density field around the sphere for an inviscid simulation: (a) Case 1, $t = 8.66 \times 10^{-6}$s; (b) Case 1, $t = 5.47 \times 10^{-5}$s. "+" denotes the experimental correlation.

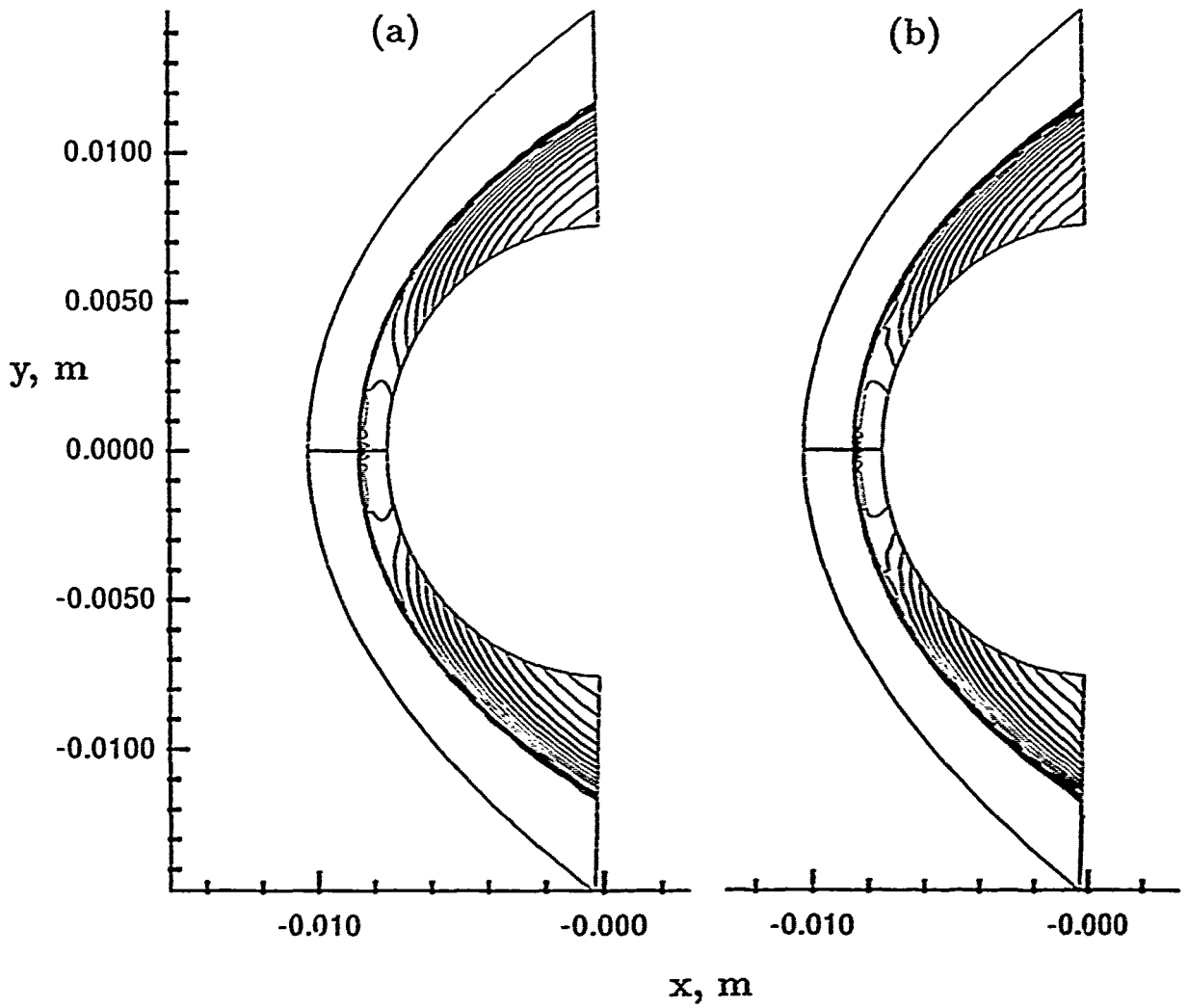


Figure 24: Density field around the sphere for two viscous simulations with tangency boundary conditions: (a) Case 2, $t = 3.76 \times 10^{-5}$s, first-order interpolation; (b) Case 3, $t = 4.01 \times 10^{-5}$s, high-order MUSCL interpolation.

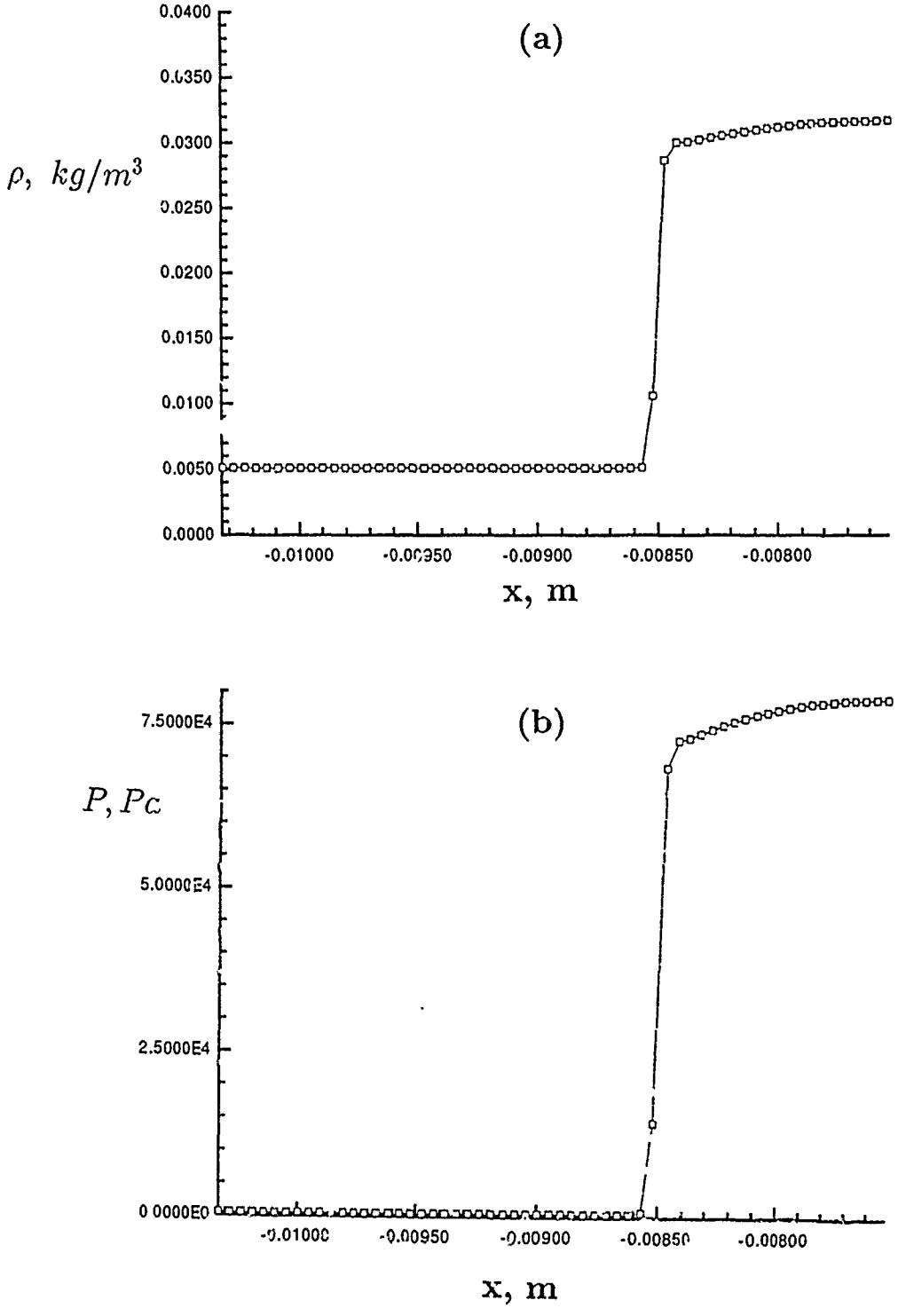


Figure 25: Flow properties for the cells adjacent to the ($y = 0$) stagnation line for case 3: (a) density; (b) pressure.

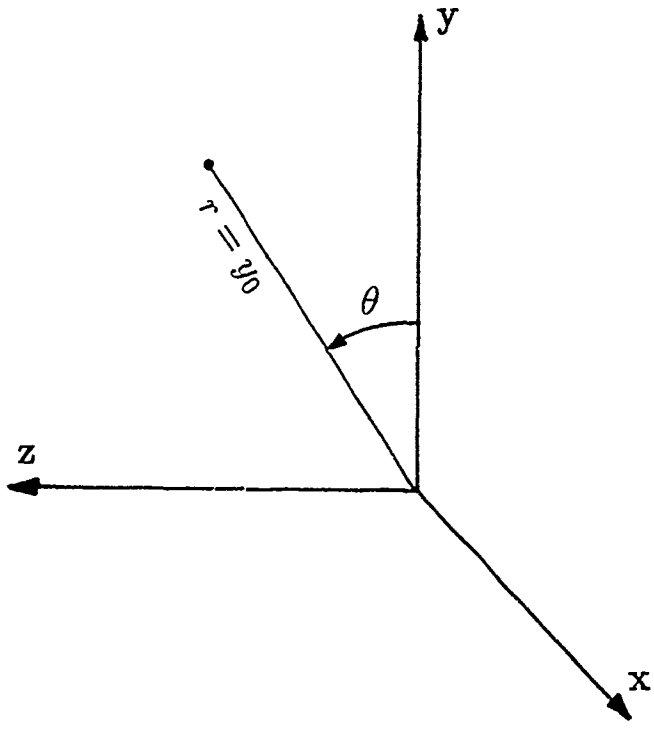


Figure 26: Cylindrical- and cartesian-coordinates.

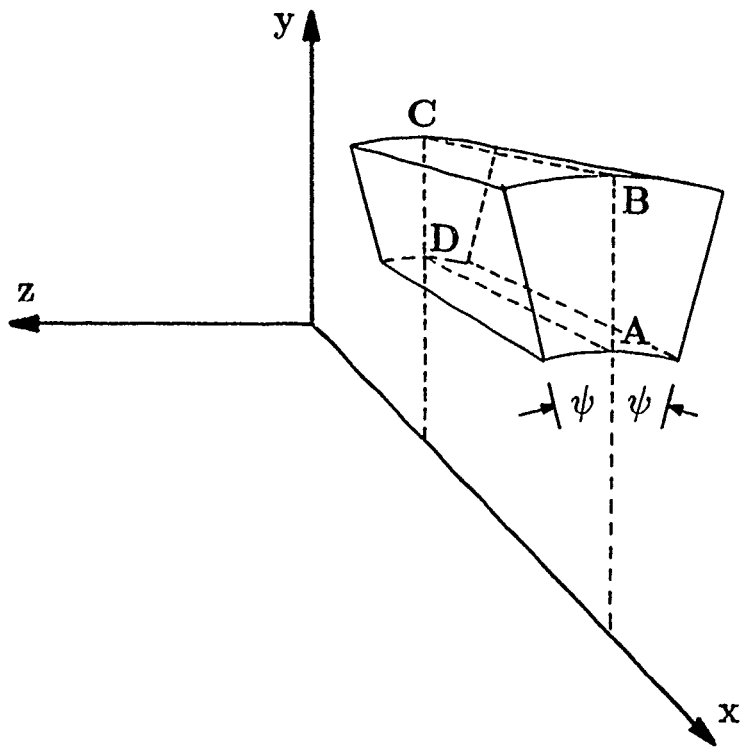


Figure 27: Axisymmetric finite-volume cell.

NASA National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA CR-187613 ICASE Interim Report 18		

4. Title and Subtitle	5. Report Date
SINGLE-BLOCK NAVIER-STOKES INTEGRATOR	July 1991
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
P. A. Jacobs	Interim Report No. 18
	10. Work Unit No
	505-90-52-01

9. Performing Organization Name and Address	11. Contract or Grant No.
Institute for Computer Applications in Science and Engineering Mail Stop 132C, NASA Langley Research Center Hampton, VA 23665-5225	NAS1-18605
	13. Type of Report and Period Covered
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Langley Research Center Hampton, VA 23665-5225	Contractor Report
	14. Sponsoring Agency Code

15. Supplementary Notes

Langley Technical Monitor:
Michael F. Card

Final Report

16. Abstract

This report describes a program for the time-integration of the Navier-Stokes equations on a two-dimensional structured mesh. The flow geometry may be either planar or axisymmetric. The unusual features of this program are that it is written in C and makes extensive use of sophisticated data structures to encapsulate the data. The idea of writing the code this way is to make it easier (than traditional FORTRAN codes) to "parallelize" for the Multiple-Instruction-Multiple-Data style of parallel computer.

The integral form of the governing equations are given for cartesian coordinates and then the particular discretization used in the code is described. A derivation of the axisymmetric equations is given in an appendix. The full version of the code describes a flow domain as a set of abutting blocks, each consisting of a *tensor-product* mesh of quadrilateral cells. However, this report considers only the single-block version of the code. The flow field is recorded as cell-average values at cell centres and explicit time stepping is used to update conserved quantities. MUSCL-type interpolation and a three-stage Riemann solver are used to calculate inviscid fluxes across cell faces while central differences (via the divergence theorem) are used to calculate the viscous fluxes. The Riemann solver is suitable for flows with very strong shocks and does not require the entropy fix as applied to the Roe-type solvers. Because the code is intended to be a test-bed for implementation on parallel computers, the coding details are described in some detail.

A set of test problems is also included. These exercise various parts of the code and should be useful for both validation and performance measurements of the (future) parallel implementations.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Navier-Stokes, Numerical Simulation, Finite-Volume, Riemann Solver	34 - Fluid Mechanics and Heat Transfer 64 - Numerical Analysis Unclassified - Unlimited

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price
Unclassified	Unclassified	71	A04

NASA FORM 1626 OCT 86